A SCALABLE PARALLEL FINITE ELEMENT FRAMEWORK FOR GROWING GEOMETRIES. APPLICATION TO METAL ADDITIVE MANUFACTURING.

ERIC NEIVA, SANTIAGO BADIA, ALBERTO F. MARTÍN, AND MICHELE CHIUMENTI

Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain.

Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Building C1, Campus Nord UPC, Gran Capitán S/N, 08034 Barcelona, Spain.

{eneiva,sbadia,amartin,michele}@cimne.upc.edu.

Abstract. *This work introduces an innovative parallel, fully-distributed finite element framework for growing geometries and its application to metal additive manufacturing. It is well-known that virtual part design and qualification in additive manufacturing requires highly-accurate multiscale and multiphysics analyses. Only high performance computing tools are able to handle such complexity in time frames compatible with time-to-market. However, efficiency, without loss of accuracy, has rarely held the centre stage in the numerical community. Here, in contrast, the framework is designed to adequately exploit the resources of high-end distributed-memory machines. It is grounded on three building blocks: (1) Hierarchical adaptive mesh refinement with octree-based meshes; (2) a parallel strategy to model the growth of the geometry; (3) state-of-the-art parallel iterative linear solvers. Computational experiments consider the heat transfer analysis at the part scale of the printing process by powder-bed technologies. After verification against a 3D benchmark, a strong-scaling analysis assesses performance and identifies major sources of parallel overhead. A third numerical example examines the efficiency and robustness of (2) in a curved 3D shape. Unprecedented parallelism and scalability were achieved in this work. Hence, this framework contributes to take on higher complexity and/or accuracy, not only of part-scale simulations of metal or polymer additive manufacturing, but also in welding, sedimentation, atherosclerosis, or any other physical problem where the physical domain of interest grows in time.*

Keywords: Parallel computing, domain decomposition, finite elements, adaptive mesh refinement, additive manufacturing, powder-bed fusion.

1. Introduction

Additive Manufacturing (AM), broadly known as 3D Printing, is introducing a disruptive design paradigm in the manufacturing landscape. The key potential of AM is the ability to cost-effectively create *on-demand* objects with complex shapes and enhanced properties, that are near impossible or impractical to produce with conventional technologies, such as casting or forging. Adoption of AM is undergoing an exponential growth lead by the aerospace, defence, medical and dental industries and the prospect is a stronger and wider presence as a manufacturing technology [70].

Nowadays, one of the main showstoppers in the AM industry, especially for metals, is the lack of a software ecosystem supporting fast and reliable product and process design. Part qualification is chiefly based on slow and expensive trial-and-error physical experimentation and the understanding of the process-structure-performance link is still very obscure. This situation precludes further implementation of AM and it is a call to action to shift to a virtual-based design model, based on predictive computer simulation tools. Only then will it be possible to fully leverage the geometrical freedom, cost efficiency and immediacy of this technology.

This work addresses the numerical simulation of metal AM processes through *High Performance Computing (HPC) tools*. The mathematical modelling of the process involves dealing with multiple scales in space (e.g. part, melt pool, microstructure), multiple scales in time (e.g. microseconds, hours),

Date: April 26, 2019.

coupled multiphysics [24, 45] (e.g. thermomechanics, phase-change, melt pool flow) and arbitrarily complex geometries that grow in time. As a result, high-fidelity analyses, which are vital for part qualification, can be extremely expensive and require vast computational resources. In this sense, HPC tools capable to run these highly accurate simulations in a time scale compatible with the time-to-market of AM are of great importance. By efficiently exploiting HPC resources with scalable methods, one can drastically reduce CPU time, allowing the optimization of the AM building process and the virtual certification of the final component in reasonable time.

Experience acquired in modelling traditional processes, such as casting or welding [17, 18, 50], has been the cornerstone of the first models for metal AM processes [2, 13, 47, 53, 62]. At the part scale, Finite Element (FE) modelling has proved to be useful to assess the influence of process parameters [59], compute temperature distributions [20, 25], or evaluate distortions and residual stresses [27, 52, 61]. Recent contributions have introduced microstructure simulations of grain growth [49, 63] and crystal plasticity [44], melt-pool-scale models [45, 72] and even multiscale and multiphysics solvers [43, 51, 64, 71, 73]. Furthermore, advanced frameworks (e.g. grounded on multi-level *hp*-FE methods combined with implicit boundary methods [46]) or applications to topology optimization [68] have also been considered.

However, in spite of the active scientific progress in the field, the authors have detected that very little effort has turned to the design of *large scale* FE methods for metal AM. Even if computational efficiency has been taken into consideration in several works, all approaches have been limited to Adaptive Mesh Refinement (AMR) [26, 60] or simplifications [21, 37, 38, 66] that sacrifice the accuracy of the model. Parallelism and scalability has been generally disregarded (with few exceptions [49, 57]), even if it is fundamental to face more complexity and/or provide increased accuracy at acceptable CPU times. For instance, for the high-fidelity melt-pool solver in [72], a simulation of 16 ms of physical time with 7 million cells requires 700 h of CPU time on a common desktop with an Intel Core i7-2600.

The purpose of this work is to design a novel scalable parallel FE framework for metal AM *at the part scale*. Our approach considers three main building blocks:

(1) *Hierarchical AMR with octree meshes* (see Sect. 2). The dimensions of a part are in the order of [mm] or [cm], but relevant physical phenomena tend to concentrate around the melt pool [μm]. Likewise, in powder-bed fusion, the layer size is also [μm], i.e. the scale of growth is much smaller than the scale of the part. Hence, adaptive meshing can be suitably exploited for the highly-localized nature of the problem. Here, a parallel octree-based *h*-adaptive FE framework [7] is established.

(2) *Modelling the growth of the geometry* (see Sect. 3). In welding and AM processes, the addition of material into the part has been typically modelled by adding new elements into the computational mesh. To this effect, the simulation starts with the generation of a *static* background mesh comprising the substrate and the filling, i.e. the final part. Common practice in the literature essentially considers two different techniques [19, 55]: *quiet*-element method and Element-Birth Method (EBM). The only difference among them is how they treat the elements in the filling at the start of the simulation. While the former assigns to them penalized material properties, perturbing the original problem; in the latter, they have no Degrees of Freedom (DOFs). At each time step, the computational mesh is updated: elements inside the incremental growth region are found with a search operation and assigned the usual material properties or new degrees of freedom, respectively. This work extends the EBM to (1). The parallelization approach is designed such that it can be accommodated in a general-purpose FE code. It only requires two interprocessor communications and it is completed with an efficient and embarrassingly parallel intersection test *for rectangular bodies* to drive the update of the computational mesh (Sect. 3.2). Finally, a strategy is devised to balance the computational load among processors, during the growth of the geometry (Sect. 3.3). The parallel and adaptive EBM is central to a parallel FE framework for growing domains and constitutes the main novelty of this work.

(3) *State-of-the-art parallel iterative linear solvers.* Compared to sparse direct solvers, iterative solvers can be efficiently implemented in parallel computer codes for distributed-memory machines. However, they must also be equipped with efficient preconditioning schemes, i.e. preconditioners able to keep a bound of the condition number, independent of mesh resolution, while still exposing a high degree of parallelism. Examples of preconditioners that the framework is able to use include Algebraic MultiGrid (AMG) [22] or Balancing Domain Decomposition by Constraints (BDDC) [6], although they are not actually exploited in this work, for reasons made clear in Sect. 4, related to the application problem.

As the originality of the framework centres upon the computational aspects to efficiently deal with growing geometries, (1) and (2) are presented in an abstract way, i.e. without considering a reference physical problem. Afterwards, Sect. 4 considers an application to the heat transfer analysis of metal AM processes by powder-bed fusion. Nonetheless, the authors believe that the framework can readily be extended to a coupled thermomechanical analysis, as long as proper treatment of history variables within the AMR framework can be guaranteed. Likewise, it may also be adapted to other metal or polymer technologies, or even be useful to other domains of study, such as the simulation of sedimentation processes.

Computer implementation was supported by `FEMPAR` [8, 9], a general-purpose object-oriented multithreaded/message-passing scientific software for the fast solution of multiphysics problems governed by Partial Differential Equations (PDEs). `FEMPAR` adapts to a range of computing environments, from desktop and laptop computers to the most advanced HPC clusters and supercomputers. The `FEMPAR-AM` module for FE analyses of metal AM processes has been developed on top of this high-end infrastructure. Its main software abstractions are described in Sect. 5. The exposition is intended to help in the customization of any general-purpose FE code for growing domains.

The numerical study of the framework in Sect. 6 starts with a verification of the thermal FE model against a well-known 3D benchmark. Validation of this heat transfer formulation has already been object of previous works [20, 21] and it is not covered here. A strong-scaling analysis follows in Sect. 6.2 to analyse the performance of the computer implementation and expose sources of load imbalance, identified as a major parallel overhead threatening the efficiency of the implementation. The simulation considers the printing of 48 layers in a cuboid, one layer printed per time step (followed by an interlayer cooling step). A relevant outcome is the capability of simulating the printing and cooling of a single layer (two linearized time steps) in a 10 million unknown problem in merely 2.5 seconds average with 6,144 processors. The last example in Sect. 6.3 considers a curved 3D shape and follows the actual laser path point-to-point. A second order adaptive mesh with no geometrical error is transformed during the simulation to accommodate the laser path. Regardless of having nonrectangular cells, the parallel EBM is capable of tracking the laser path, as long as some quasi-rectangularity conditions hold. In conclusion (see Sect. 7), the fully-distributed, parallel framework presented in this work is set to contribute to the efficient simulation of AM processes, a critical aspect long identified, though mostly neglected by the numerical community.

2. Mesh generation by hierarchical AMR

Physical phenomena are often characterized by multiple scales in both space and time. When the smallest ones are highly-localized in the physical domain of analysis, uniformly refined meshes tend to be impractical from the computational viewpoint, even for the largest available supercomputers.

The purpose of AMR is to reach a compromise between the high-accuracy requirements in the regions of interest and the computational effort of solving for the whole system. To this end, the mesh is refined in the regions of the domain that present a complex behaviour of the solution, while larger mesh sizes are prescribed in other areas.

In this work, the areas of interest are known *a priori* and correspond to the growing regions. It is assumed that the geometrical scale of growth is much smaller than the domain of study, as it is the case of welding or AM processes. Besides, the framework is restricted to h-adaptivity, i.e. only the mesh size changes among cells, in contrast to hp-adaptivity, where the polynomial order p of the FEs

may also vary among cells. This computational framework is briefly outlined in this section; the reader is referred to [7] for a thorough exposition.

2.1. **Hierarchical AMR with octree meshes.** Let us suppose that $\Omega \subset \mathbb{R}^d$ is an open bounded polyhedral domain, being $d = 2, 3$ the dimension of the physical space. Let $\mathcal{T}_h^0$ be a conforming and quasi-uniform partition of Ω into quadrilaterals ($d = 2$) or hexahedra ($d = 3$), where every $K \in \mathcal{T}_h^0$ is the image of a reference element $\hat{K}$ through a smooth bijective mapping F_K. If not stated otherwise, these hypotheses are common to all sections of this document.

Hierarchical AMR is a multi-step process. The mesh generation consists in the transformation of $\mathcal{T}_h^0$, typically as simple as a single quadrilateral or hexahedron, into an objective mesh $\mathcal{T}_h$ via a finite number of refinement/coarsening steps; in other words, the AMR process generates a sequence $\mathcal{T}_h^0, \mathcal{T}_h^1, \ldots, \mathcal{T}_h^m \equiv \mathcal{T}_h$ such that $\mathcal{T}_h^i = \mathcal{R}(\mathcal{T}_h^{i-1}, \theta^i)$, $i = 1, \ldots, m < \infty$, where $\mathcal{R}$ applies the refinement/coarsening procedure over $\mathcal{T}_h^{i-1}$ and $\theta^i : \mathcal{T}_h^{i-1} \to \{-1, 0, 1\}$ is an array establishing the action to be taken at each cell: -1 for coarsening, 0 for "do nothing" and 1 for refinement.

A cell marked for refinement is partitioned into four (2D) or eight (3D) children cells by bisecting all cell edges. As a result, $\mathcal{T}_h$ can be interpreted as a collection of quadtrees (2D) or octrees (3D), where the cells of $\mathcal{T}_h^0$ are the roots of these trees, and children cells (a.k.a. quadrants or octants) branch off their parent cells. The leaf cells in this hierarchy form the mesh in the usual meaning, i.e. $\mathcal{T}_h$. Furthermore, for any cell $K \in \mathcal{T}_h$, $l(K)$ is the *level* of refinement of K in the aforementioned hierarchy. In particular, $l(K) = 0$ for the root cells and $l(K) = l(\text{parent(K)}) + 1$, otherwise. The level can also be defined for lower dimensional mesh entities (vertices, edges, faces) as in [39, Def. 2.4]. Fig. 1 illustrates this recursive tree structure with cells at different levels of refinement stemming from a single root.

Octree meshes admit a very compact computer representation, based on Morton encoding [56] by bit interleaving, which enables efficient manipulation in high-end distributed-memory computers [14]. Moreover, they provide multi-resolution capability by local adaptation, as the leaves in the hierarchy can be at different levels of refinement. However, they are potentially nonconforming by construction, e.g. there can be *hanging* vertices in the middle of an edge or face or *hanging* edges or faces in touch with a coarser geometrical entity.

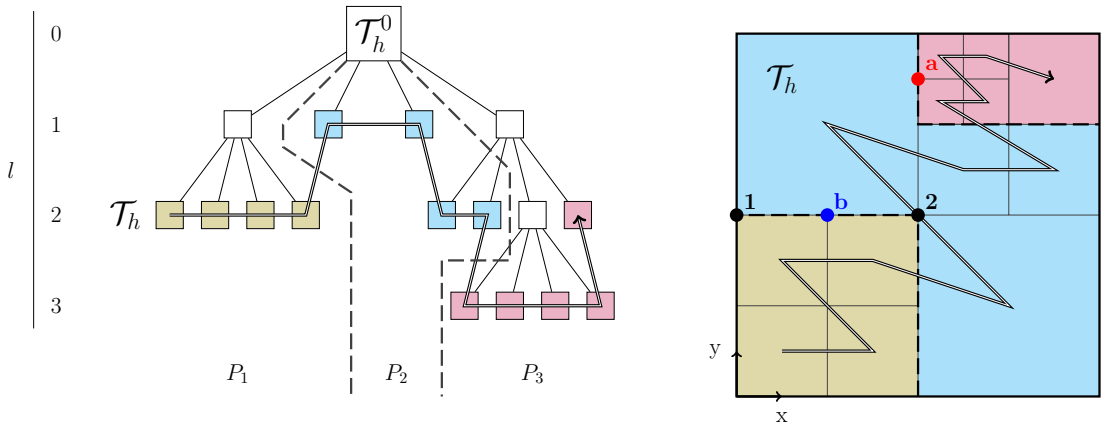


FIGURE 1. The hierarchical construction of $\mathcal{T}_h$ gives rise to a one-to-one correspondence between the cells of $\mathcal{T}_h$ and the leaves of a quadtree rooted in $\mathcal{T}_h^0$. $\mathcal{T}_h$ does not satisfy the *2:1 balance*, e.g. the red hanging vertex a is not permitted. Assuming a discretization with conforming Lagrangian linear FEs, the value of the DOF at hanging vertex b is subject to the constraint $V_b = 0.5V_1 + 0.5V_2$. $\mathcal{T}_h$ is partitioned into three subdomains, $P1$, $P2$ and $P3$, using the z-order curve obtained by traversing the leaves of the octree in increasing Morton index. Adapted from [14].

Nonconformity introduces additional complexity in the implementation of conforming FEs, especially in parallel codes for distributed-memory computers. This degree of complexity is nevertheless significantly reduced by enforcing the so-called *2:1 balance* relation, i.e. adjacent cells may differ at most by a single level of refinement. In this sense, the mesh in Fig. 1 violates the 2:1 balance, because hanging vertex a is surrounded by cells that differ by two levels.

In order to preserve the continuity of a conforming FE approximation, DOFs sitting on *hanging* geometric entities cannot have an arbitrary value, they are constrained by neighbouring nonhanging DOFs. The approach advocated in this work to handle the hanging node constraints [48] consists in eliminating them from the local matrices, before assembling the global matrix. In this case, hanging DOFs are not associated to global DOFs and, thus, a row/column in the global linear system of equations.

2.2. **Partitioning the octree mesh.** Efficient and scalable parallel partitioning schemes for adaptively refined meshes are still an active research topic. Our computational framework relies on the `p4est` library [14]. `p4est` is a Message Passing Interface (MPI) library for efficiently handling (forests of) octrees that has scaled up to hundreds of thousands of cores [11]. Using the properties of the Morton encoding, `p4est` offers, among others, parallel subroutines to refine (or coarsen) the octants of an octree, enforce the 2:1 balance ratio and partition/redistribute the octants among the available processors to keep the computational load balanced [14, 39]. Data structures and algorithms involved in the interface between `p4est` and `FEMPAR` are detailed in [7]. They are configured according to a two-layered meshing approach. The first or *inner* layer is a light-weight encoding of the forest-of-trees, handled by `p4est`. The second or *outer* layer is a richer mesh representation, suitable for generic finite elements.

The principle underlying the mesh partitioner is the use of space-filling curves. Octants within an octree can be naturally assigned an ordering by a traversal across all leaves, e.g. increasing Morton index, as shown in Fig. 1. Application of the one-to-one correspondence between tree nodes and octants reveals that this one-dimensional sequence corresponds exactly to a z-order space-filling curve of the triangulation $\mathcal{T}_h$. Hence, the problem of partitioning $\mathcal{T}_h$ can be reformulated into the much simpler problem of subdividing a one-dimensional curve. This circumvents the parallel scaling bottleneck that commonly arises from *dynamic load balancing* via graph partitioning algorithms [40, 41].

However, the simplicity comes at a price related to the fact that, among space-filling curves, z-curves have unbounded locality and low bounding-box quality [36]. In our context, this leads to the emergence of poorly-shaped and, possibly, disconnected subdomains (with at most two components [15] for single-octree meshes). Bad quality of subdomains affects the performance of nonoverlapping Domain Decomposition (DD) methods [48].

3. Modelling the growth of the geometry

3.1. **Parallel element-birth method.** As mentioned in Sect. 1, the growth of the geometry is modelled in a *background* FE mesh that, even if it is refined or coarsened, always covers the same domain.

Let $\Omega(t)$ be a *growing-in-time* domain. During the time interval $[t_\mathrm{i}, t_\mathrm{f}]$, $\Omega(t)$ transforms from an initial domain Ω_i to a final one Ω_f. For the sake of simplicity, AMR is not considered for now, only later in the exposition, and it is assumed that there exists (1) a *background* conforming partition $\mathcal{T}_h \equiv \mathcal{T}_h^0 = \{K\}$ of Ω_f, where Ω_f and $\mathcal{T}_h^0$ satisfy the hypotheses stated in the first paragraph of Sect. 2.1, and (2) a time discretization $t_\mathrm{i} = t_0 < t_1 < \ldots < t_{N_t} = t_\mathrm{f}$ such that, for all $j = 0, \ldots, N_t$, a partition $\mathcal{T}_{h,j}$ of $\Omega(t_j)$ can be obtained as a subset of cells of $\mathcal{T}_h$. In other words, a body-fitted mesh of Ω_f can be built so that subsets of this mesh can be taken as body-fitted meshes of $\Omega(t_j)$, $j = 0, \ldots, N_t$. As $\Omega(t)$ grows in time, the relation $\mathcal{T}_{h,\mathrm{i}} = \mathcal{T}_{h,0} \subseteq \mathcal{T}_{h,1} \subseteq \ldots \subseteq \mathcal{T}_{h,N_t} = \mathcal{T}_{h,\mathrm{f}}$ holds.

This setting is typical in welding or AM simulations. In AM, for instance, it is frequently required that the mesh of the component conforms to the layers. On the other hand, the method presented below can be adapted with little effort to a more general setting, where the growth-fitting requirement

is dismissed by resorting to unfitted or immersed boundary methods [10]. In this case, $\mathcal{T}_h$ is a triangulation of an artificial domain Ω_{art}, such that it includes the final physical domain, i.e. $\Omega_{\text{f}} \subset \Omega_{\text{art}}$, but it is also characterized by a simple geometry, easy to mesh with Cartesian grids.

Consider now partitions of $\mathcal{T}_h$ of the form $\{\mathcal{T}_{h,j}, \mathcal{T}_h \setminus \mathcal{T}_{h,j}\},\ j = 0, \ldots, N_t$. In this classification, the cells in $\mathcal{T}_{h,j}$ are referred to as the *active* cells K_{ac}, while the ones in $\mathcal{T}_h \setminus \mathcal{T}_{h,j}$ as the *inactive* cells K_{in}. The key point of the EBM is to assign degrees of freedom only in active cells, that is, the computational domain, at any $j = 0, \ldots, N_t$, is defined by $\mathcal{T}_{h,j} = \{K_{\text{ac}}\}$. In this way, inactive cells do not play any role in the numerical approximation; they have no contribution to the global linear system, in contrast with the quiet-element method [55]. Besides, note the similarities of this approach to the one employed in unfitted FE methods [5, 10], distinguishing interior and cut (active) cells from exterior (inactive) cells.

This representation of a growing domain is completed with a procedure to update the computational mesh during a time increment. The most usual approach is to use a search algorithm to find the set of cells in $\mathcal{T}_{h,j+1} \setminus \mathcal{T}_{h,j},\ j = 0, \ldots, N_t - 1$, referred to as the *activated* cells K_{acd}. Then, $\mathcal{T}_{h,j+1} = \mathcal{T}_{h,j} \cup \{K_{\text{acd}}\}$ defines the next computational mesh, as illustrated in Fig. 5. This means that $\mathcal{T}_{h,j+1}$ receives the old DOF global identifiers (and FE function DOF values) from $\mathcal{T}_{h,j}$ and has new degrees of freedom assigned in the activated cells. The initial value of these new DOFs is set with a criterion that depends on the application problem, as seen in Sect. 4.

Therefore, in a parallel distributed-memory environment, there are two different partitions of $\mathcal{T}_{h,j}$ playing a role in the simulation: (1) into subdomains $\mathcal{T}^i_{h,j},\ i = 1, \ldots, n^{\text{sbd}}$ and (2) into active K_{ac} and inactive K_{in} cells. Assuming a one-to-one mapping among subdomains and CPU cores, in our approach [7], each processor stores the geometrical information corresponding to the subdomain portion $\mathcal{T}^i_{h,j}$ of the global mesh and one layer of off-processor cells surrounding the local subdomain mesh, the so-called *ghost* cells. With regards to (2), each cell is associated a *status*, e.g. an integer value, that expresses whether it is an active or inactive cell. It follows that $\mathcal{T}^i_{h,j}$ can be composed of both *active* and *inactive* cells.

As explained, the update of (2) follows from finding the subset K_{acd}. *Local* activated cells, i.e. $\mathcal{T}^i_{h,j} \cap \{K_{\text{acd}}\}$, are found with the search algorithm. Afterwards, a nearest-neighbour communication is carried out to update the status at the ghost cells. The second step is necessary to know whether a face sitting on the interprocessor contour is in the interior or at the boundary of the domain $\mathcal{T}_{h,j+1}$.

Bringing now briefly AMR into the discussion, some considerations with regards to the EBM must be taken into account when applying mesh transformations. First, when refining a cell, its status is inherited by the child cells. A cell can only be coarsened, when all siblings have the same status, otherwise the computational domain is perturbed. Finally, after a partition/redistribution of cells across processors, the status of the redistributed cells must be migrated with interprocessor communication. Adding this to the update of the status at ghost cells, the EBM in a parallel AMR framework only demands two extra interprocessor data transfers with respect to a standard FE simulation pipeline.

3.2. **Parallel search algorithm.** The update of the computational mesh consists in finding the cells in the mesh that are inside the *known* growing region of the current time increment. In this sense, the problem is a standard and well known collision detection that can be tackled with any of the many existing algorithms. Our goal is to derive from them a strategy that is both computationally inexpensive and highly-parallelizable for octree-based meshes.

The approach adopted in this work is founded on the Hyperplane Separation Theorem (HST) [12]. It states that given A and B two disjoint, nonempty and *convex* subsets of $\mathbb{R}^n$, there exists a nonzero vector v and a real number c, such that $\langle x, v\rangle \geq c$ and $\langle y, v\rangle \leq c$, for any $x \in A,\ y \in B$. In other words, the hyperplane $\langle \cdot, v\rangle = c$, with v its normal vector, separates A and B. A corollary of this theorem is that, if A and B are convex *polyhedra*, possible separating planes are either parallel to a face of one of the polyhedra or contain an edge from each of the polyhedra.

In our context, assuming the FE mesh is formed by rectangular hexahedra and the search volume is a cuboid, as in Sect. 4, the purpose is to test the intersection between two cuboids (any mesh cell vs search volume). In this case, the HST narrows down the number of potential separating planes to

fifteen [28, 34]: three for the independent faces of the cell, three for the independent faces of the search cuboid and nine generated by all possible independent pairs formed by an edge from the cell and an edge from the search cuboid. It follows that two cuboids intersect, if and only if, none of the fifteen possible separating planes exists.

A separating plane can be tested by comparing the projections of the cuboids onto a line perpendicular to the plane, referred to as the separating axis (Fig. 2). If the intervals of the projections do not intersect, then the cuboids do not intersect themselves. In [28], it is shown how this test amounts to compare two real quantities that depend on the dimensions and unit directions of the cell, the dimensions and unit directions of the search cuboid and the vector joining the centroids of the two cuboids. For each separating axis, the nonintersection test in terms of these quantities is given in [28, Tab. 1] or [35, Sect. 4.6].

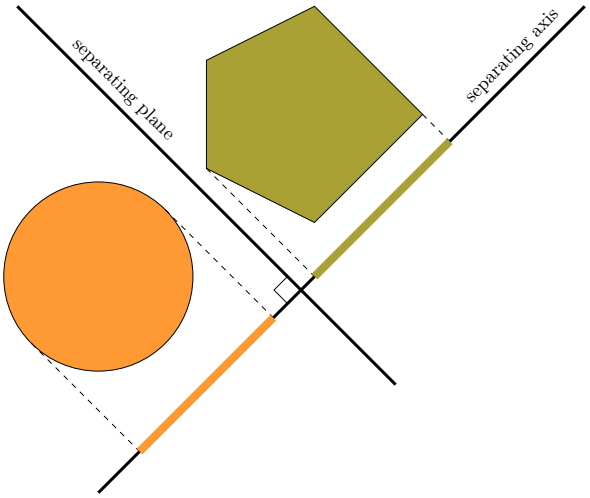
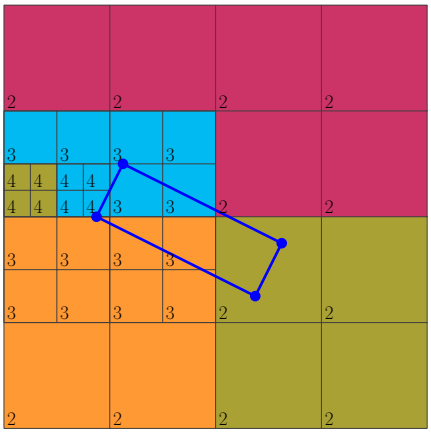


FIGURE 2. Illustration of the HST. A separating plane can be tested by examining whether the projections of the two convex bodies onto a line perpendicular to the plane intersect or not.

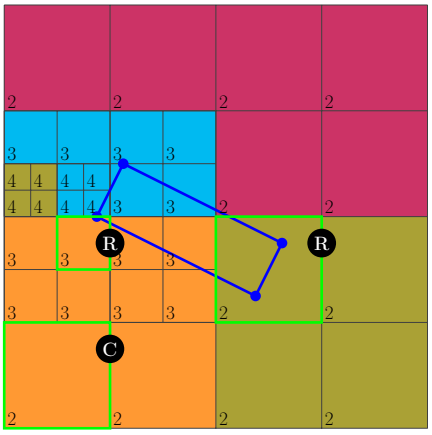
At this point, it remains to see how this test can be exploited for the parallel search algorithm with adaptive octree-based meshes. Dropping the assumption in Sect. 3.1 of sequential mesh inclusion, i.e. $\mathcal{T}_{h,j} \neq \mathcal{T}_{h,j+1}, \forall j = 0, \ldots, N_t$, now $\mathcal{T}_{h,j}$ is transformed into $\mathcal{T}_{h,j+1}$, by applying several refinement (coarsening) operations to the octants intersecting (nonintersecting) the search cuboid of the $j \to j+1$ time increment. The transformation finishes when all octants intersecting the search cuboid have a given maximum level of refinement. In fact, these octants form precisely the subset K_{acd}. The number of transformations required is problem-dependent, but it is upper-bounded by the difference between the user-prescribed maximum and minimum levels of refinement.

Therefore, the mesh transformation from $\mathcal{T}_{h,j}$ into $\mathcal{T}_{h,j+1}$ is carried out in a finite number of refinement/coarsening steps, each one determined by a cell-wise search. Specifically, the criterion to decide whether an octant is refined or coarsened is to perform the nonintersection test against the search cuboid. If it passes (fails), the octant is coarsened (refined). An example of this procedure is shown in Fig. 3. As observed, if all processors know the dimensions of the search cuboid, the algorithm is embarrassingly parallel, in particular, it does not require interprocessor communication. By construction of `p4est` a subdomain can only prescribe refinement/coarsening operations to its own local cells. It follows that the nonintersection test on ghost cells is redundant; the status of ghost cells must be updated at the end of the mesh transformation with a nearest-neighbour communication.

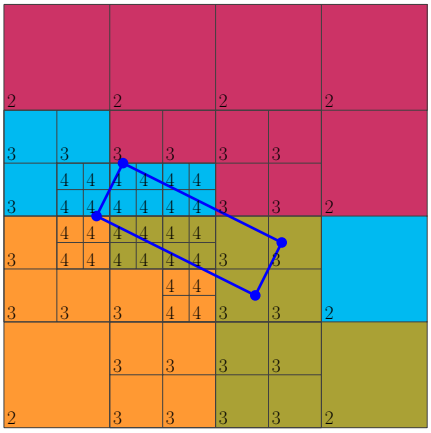
The search algorithm can be further accelerated by intersecting beforehand the search cuboid (or a bounding box of it) against the subdomain limits. In this case, the procedure can be skipped on



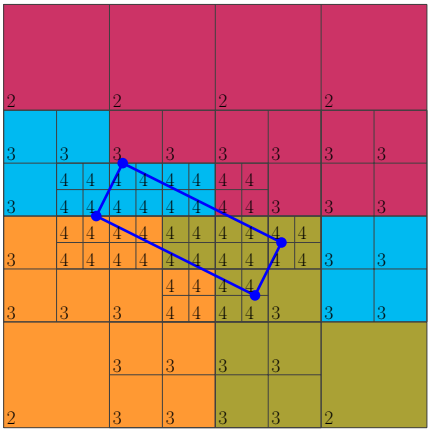
(A) Given $\mathcal{T}_{h,j}, j = 0, \ldots, N_t$, compute the search volume for the time increment $\Delta t^{j \to j+1}$.



(B) Loop over cells in $\mathcal{T}_{h,j}$. If nonintersection test fails (passes), then mark cell for refinement (coarsening). Some examples are highlighted (R = refine, C = coarsen).



(C) Refine, coarsen and redistribute cells.



(D) Repeat (B)-(C) until all cells with maximum level of refinement that intersect the cuboid have been found.

FIGURE 3. 2D example illustrating the iterative procedure to transform $\mathcal{T}_{h,j}$ into $\mathcal{T}_{h,j+1}$. The maximum and minimum levels of refinement are 4 and 2. The level of each cell is written at their lower left corner. Each colour represents a different subdomain. Note that, from one step to the next one, some cells that do not intersect the search volume have to be refined to keep the 2:1 balance.

those subdomains that do not intersect the cuboid. On the other hand, faster tests can be designed for uniformly refined meshes, such as checking whether the centroid of the cell is inside the search cuboid. Note that this test is not equivalent to the method of separating axes. Besides, it is not suitable for octree meshes. For instance, it may not transform the mesh at all if the search cuboid sits on top of heavily coarsened cells.

Apart from that, the algorithm is limited to rectangular meshes and search volumes. In more general cases, e.g. high-order meshes, the hypothesis of convexity may not hold; i.e. the hyperplane separation theorem cannot be the starting point; a more general method must be adopted instead. However, as shown in the example of Sect. 6.3, a high order mesh can be configured such that the search cuboid always overlaps a region of the mesh, with enough resolution to assume its cells are quasi-rectangular.

3.3. **Dynamic load balancing.** When designing a scalable application, the partition into subdomains must be defined such that it evenly distributes among processors the total computational load. However, to this goal, the EBM adds two mutually excluding constraints; indeed, while the size of data structures and the complexity of procedures that manipulate the mesh grow with the total number of cells, those concerning the FE space, the FE system and the linear solver depend on the number of active cells (i.e. number of DOFs).

The distribution of computational work can be tuned by allowing for a user-specified *weight function* w that assigns a non-negative integer value to each octant. The partition can then be constructed by equally distributing the accumulated weights of the octants that each processor owns, instead of the number of owned octants per processor. As `p4est` provides such capability (see [14, Sect. 3.3]), the remaining question is to decide how to define w, taking into account the constraints above.

The answer depends on how the computational time is distributed among the different stages of the FE simulation pipeline. In the context of growing domains, FE analysis is a long transient and it may be often desirable to reuse the same mesh for several time steps, seeking to minimize the AMR events and, thus, reduce simulation times. In this scenario, the number of time steps (linear system solutions) is greater than the number of mesh transformations and w should favour the balance of active cells. With this idea in mind, the weight function can be defined as

$$w_K = \begin{cases} w_a & \text{if } K \in \{K_{\text{ac}}\} \\ w_i & \text{if } K \in \{K_{\text{in}}\} \end{cases} \tag{1}$$

where $w_a \in \mathbb{N}$ and $w_i \in \mathbb{N}^0$.

The effect of this weight function is illustrated in Fig. 4. A uniform distribution of the octree octants, $(w_a, w_i) = (1, 1)$, may lead to high load imbalance in the number of DOFs per subdomain. There can even be fully inactive parts, as shown in Fig. 4(a), leaving the processors in charge of them mostly idling during local integration, assembly and solve phases. In contrast, $(w_a, w_i) = (1, 0)$ gives the most uniform distribution of $\{K_{\text{ac}}\}$, but it can also lead to extreme imbalance of $\{K_{\text{in}}\}$ and, thus, the whole set of triangulation cells. Alternatively, pairs (w_a, w_i) satisfying $w_a \gg w_i$ offer good compromise partitions.

4. Application to metal AM

4.1. **Heat transfer analysis.** After introducing the ingredients of the parallel FE framework for growing geometries, the purpose now is to apply it to the thermal analysis of an additive manufacturing process by powder-bed fusion, such as Direct Metal Laser Sintering (DMLS). This manufacturing technology is illustrated in [21, Fig. 1]. This will be the reference problem for the subsequent analysis with numerical experiments.

Let $\Omega(t)$ be a *growing* domain in $\mathbb{R}^3$ as in Sect. 3. Here, $\Omega(t)$ represents the component to be printed. The governing equation to find the temperature distribution u in time is the balance of energy equation, expressed as

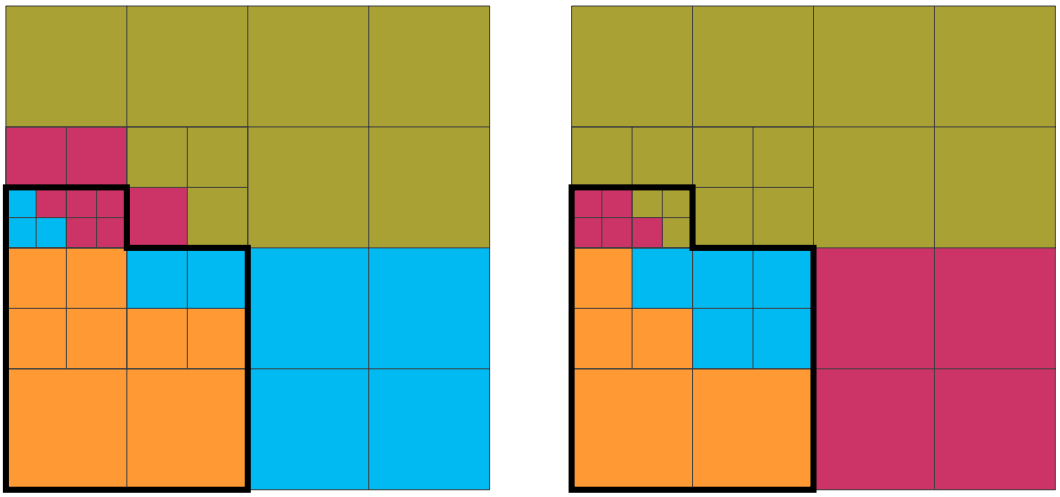
$$C(u)\partial_t u - \boldsymbol{\nabla} \cdot (k(u)\, \nabla u) = f, \quad \text{in } \Omega(t), \quad t \in [t_{\text{i}}, t_{\text{f}}], \tag{2}$$

where $C(u)$ is the heat capacity coefficient, $k(u) \geq 0$ is the thermal conductivity and f is the rate of energy supplied to the system per unit volume by the moving laser. $C(u)$ is given by the product of the density of the material $\rho(u)$ and the specific heat $c(u)$, but one may consider a modified heat capacity coefficient to also account for phase change effects [20] or compute $C(u)$ with the CALPHAD approach [42, 43, 67].

Eq. (2) is subject to the initial condition

$$u(\boldsymbol{x}, t_{\text{i}}) = u^0(\boldsymbol{x}) \tag{3}$$

and the boundary conditions [21, Fig. 2] are (1) heat conduction through the building platform, (2) heat conduction through the powder-bed and (3) heat convection and radiation through the free



(A) A default partition, that is, $(w_a, w_i) = (1, 1)$, can result in a poor balancing of DOFs and even fully inactive parts (e.g. green subdomain).

(B) By setting partition weights to, e.g. $(w_a, w_i) = (10, 1)$, the active cells can be balanced, leading to a more equilibrated parallel distribution of the DOFs.

FIGURE 4. 2D example illustrating how partition weights can be used to balance dynamically the DOFs across the processors. Each colour represents a different subdomain. Active cells are enclosed by a thick contour polygon, representing the computational domain.

surface. After linearising the Stefan-Boltzmann's law for heat radiation [20], all heat loss boundary conditions admit a unified expression in terms of Newton's law of cooling:

$$q_{\text{loss}}(u,t) = h_{\text{loss}}(u)(u - u_{\text{loss}}(t)), \;\; \text{in} \;\; \partial\Omega^{\text{loss}}(t), \;\; t \in [t_{\text{i}}, t_{\text{f}}], \tag{4}$$

where *loss* refers to the kind of heat loss mechanism (conduction through solid, conduction through powder or convection and radiation) and the boundary region where it applies (contact with building platform, interface solid-powder or free surface).

The weak form of the problem defined by Eqs. (2)-(4) can be stated as: *Find* $u(t) \in \mathcal{V}_t = \mathrm{H}^1(\Omega(t))$, *almost everywhere in* $(t_{\text{i}}, t_{\text{f}}]$, *such that*

$$(C(u)\partial_t u, v) - (k(u)\nabla u, \nabla v) + \langle h_{\text{loss}}(u)u, v\rangle_{\partial\Omega^{\text{loss}}} = \langle f, v\rangle + \langle h_{\text{loss}}(u)u_{\text{loss}}, v\rangle_{\partial\Omega^{\text{loss}}}, \;\; \forall v \in \mathcal{V}_t. \tag{5}$$

Considering now $\mathcal{T}_h$ the triangulation of Ω_{f}, $\mathrm{V}_h \subset \mathcal{V}_{t_{\text{f}}}$ a conforming FE space for the temperature field and $\{\varphi_j(\boldsymbol{x})\}_{j=1}^{N_h}$ a FE basis of the space V_h, the semi-discrete form of Eq. (5), after discretization in space with the Galerkin method and integration in time, e.g. with the semi-implicit backward Euler method, reads

$$\begin{gathered}\left[\frac{\mathbf{M}_C^n}{\Delta t^{n+1}} + \mathbf{A}^n + \mathbf{M}_{\text{loss}}^n\right]\mathbf{U}^{n+1} = \mathbf{b}_f^{n+1} + \frac{\mathbf{M}_C^n}{\Delta t^{n+1}}\mathbf{U}^n + \mathbf{b}_{\text{loss}}^n, \\ \mathbf{U}(0) = \mathbf{U}^0,\end{gathered} \tag{6}$$

where the time interval of interest $[t_{\text{i}}, t_{\text{f}}]$ has been divided in subintervals $t_{\text{i}} = t_0 < t_1 < \ldots < t_{N_t} = t_{\text{f}}$ with $\Delta t^{n+1} = t_{n+1} - t_n$ variable for $n = 0, \ldots, N_t - 1$. As a result of using the EBM (Sect. 3), Eq. (6) is only constructed in $\Omega(t_n)$, $n = 0, \ldots, N_t - 1$. In other words, local integration and assembly of Eq. (5) is only carried out at the subset of active elements $\{K_{\text{ac}}\} = \mathcal{T}_{h,n}$.

Moreover, $\mathbf{U}(t) = (\mathrm{U}_j(t))_{j=1}^{N_h}$ and $\mathbf{U}^0 = (\mathrm{U}_j^0)_{j=1}^{N_h}$ are the components of $\mathrm{U}_h(t)$ and U_h^i with respect to the basis $\{\varphi_j(\boldsymbol{x})\}_{j=1}^{N_h}$, and the coefficients of $\mathbf{M}_\square$, $\mathbf{A}$ and $\mathbf{b}_\square$ are given by:

$$\mathrm{M}_C^{n,ij} = (C(\mathbf{U}^n)\varphi_i, \varphi_j)_{\Omega(t_n)}, \quad \mathrm{M}_{\mathrm{loss}}^{n,ij} = \langle h_{\mathrm{loss}}(\mathbf{U}^n)\varphi_i, \varphi_j\rangle_{\partial\Omega^{\mathrm{loss}}\cap\partial\Omega(t_n)}, \quad \mathrm{A}^{n,ij} = (k(\mathbf{U}^n)\nabla\varphi_i, \nabla\varphi_j)_{\Omega(t_n)}$$
$$\mathbf{b}_f^{n+1,i} = \langle\varphi_i, f^{n+1}\rangle_{\Omega(t_n)}, \quad \mathbf{b}_{\mathrm{loss}}^{n,i} = \langle h_{\mathrm{loss}}(\mathbf{U}^n)\varphi_i, \mathrm{U}_{\mathrm{loss}}(t^{n+1})\rangle_{\partial\Omega^{\mathrm{loss}}\cap\partial\Omega(t_n)}.$$

An important characteristic of the physical problem is that, due to high heat capacity of metals and small time steps necessary to meet accuracy requirements, Eq. (6) is often a mass-dominated linear system. As a result, Jacobi (or diagonal) preconditioning is adopted in the numerical examples of Sect. 6. Although this preconditioner does not alter the asymptotic behaviour of the condition number of the conductivity matrix ($\mathcal{O}(h^{-2})$), it corrects relative scales of Eq. (6) arising from the fact that meshes resulting from (1) have cells at very different refinement levels, i.e. highly varying sizes.

4.2. **FE modelling of the moving thermal load.** As pointed out in Eq. (2), f is a moving thermal load that models the action of the laser in the system. But the moving heat source also drives the growth of the geometry in time, as the sintering process triggered by the laser transforms the metal powder into new solid material.

Therefore, the FE modelling of the printing process requires a method to apply the volumetric heat source f in space and time and track the growing $\Omega(t)$. The EBM presented in Sect. 3.1 can serve both purposes, as seen in Fig. 5. In this case, the set of activated cells K_{acd}, representing the incremental growth region, is also affected by the laser during the time increment, i.e. the heat source term is also integrated in these cells.

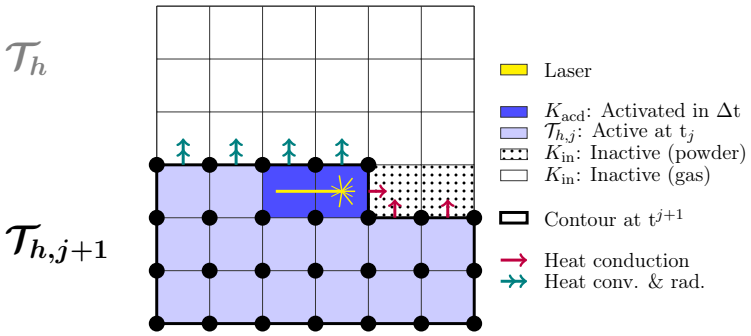


FIGURE 5. Illustration of the *element-birth* method applied to the thermal simulation of an AM process. As shown with this 2D FE cartesian grid, for any $j = 0, \ldots, N_t$, DOFs are only assigned to the set of active cells $\{K_{\mathrm{ac}}\} = \mathcal{T}_{h,j+1}$. A search algorithm is employed to identify the set of activated cells $\{K_{\mathrm{acd}}\} = \mathcal{T}_{h,j+1} \setminus \mathcal{T}_{h,j}$, where the laser is focused during Δt. The computational mesh is then updated, by assigning new DOFs in K_{acd}. Afterwards, the energy input, as stated by Eq. (7), is uniformly distributed over K_{acd}. Adapted from [21].

Another comment arises on the update of the computational mesh. In general, one aims to follow the actual path of the laser in the machine, as faithfully as possible. To this end, the search algorithm of Sect. 3.2 comes into play. The information of the laser path, together with other process parameters, such as the laser spot width, defines the search volume containing the cells affected by the energy input during the time step [20], subsequently referred to as the Heat Affected Volume (HAV).

If f is taken as a uniform heat source, the average density distribution is computed as

$$f = \frac{\eta\, W}{\mathrm{V}_{\mathrm{acd}}}, \tag{7}$$

where W is the laser power [watt], η is the heat absorption coefficient, a measure of the laser efficiency and $\mathrm{V}_{\mathrm{acd}}$ is the volume of activated cells, i.e. intersecting the HAV. Goldak-based or surface Gaussian

distributions may also be considered. In those cases, the heat source is evaluated with their corresponding analytical expressions in $\mathrm{K_{acd}}$. On the other hand, the initial temperature of the new DOFs is set to the same value as the initial value at the building platform. More accurate alternatives are analysed in the literature [55], but this aspect of the model is not relevant to the overall performance of the framework.

5. Computer implementation.

This section describes the software design of a parallel FE framework for growing domains, the so-called `FEMPAR-AM` module, atop the services provided by `FEMPAR` [8]. `FEMPAR` is an open source, general-purpose, object-oriented scientific software framework for the massively parallel FE simulation of problems governed by PDEs. `FEMPAR` software architecture is composed by a set of mathematically-supported abstractions that let its users break FE-based simulation pipelines into smaller blocks that can be used and/or extended to fit users' application requirements. Each abstraction takes charge of a different step in a typical FE simulation, including, among others, mesh generation, adaptation, and partitioning, construction of a global FE space and DOF numbering, numerical evaluation of cell and facet integrals and linear system assembly, linear system solution or generation of simulation output data for later visualization. The reader is referred to [7, 8] for a detailed exposition of the main software abstractions in `FEMPAR`. Although `FEMPAR-AM` exploits most of the software abstractions provided by `FEMPAR`, the discussion in this section mainly focuses on those which had to be particularly set up and/or customized to support growing domains. Apart from this, the section also presents newly introduced software abstractions which are particular to `FEMPAR-AM`. The exposition is intended to help the reader grasp how any general-purpose FE framework can be customized in order to deal with growing domains.

As `FEMPAR-AM` relies on the EBM (Sect. 3.1), the first software requirement that has to be fulfilled by the underlying FE framework is the ability to build global FE spaces that only carry out DOFs for cells which are active at the current simulation step. In `FEMPAR`, a global FE space is built from two main ingredients: (1) `triangulation_t` [8, Sect. 7], which represents the geometrical discretization of the computational domain into cells, and (2) `reference_fe_t` [8, Sect. 6], which represents an *abstract* local FE on top of each of the triangulation cells. A particular type extension of `reference_fe_t`, referred to as `void_reference_fe_t` [8, Sect. 6.5], implements a local FE with no degrees of freedom. The data structure in charge of building the global FE space, i.e. `fe_space_t` [8, Sect. 10], is general enough such that one may use a different local FE on different regions of the computational domain. By mapping *active* cells to standard (e.g. Lagrangian) FEs and *inactive* cells to *void* FEs, `fe_space_t` is constructed such that it assigns global DOF identifiers only to the nodes of *active* cells; while nodes surrounded by *inactive* cells do not receive a DOF identifier and, as a result, they neither assemble any contributions from local integration nor form part of the global linear system. On the other hand, the `triangulation_t` software abstraction lets its users exploit the so-called `set_id` [8, Sect. 7.1], a cell-based integer attribute. Each cell of the triangulation can be assigned a `set_id` number to group the cells into different subsets. In our case, this variable is used to store the current status of the cell in the mesh and, by `fe_space_t`, to determine which `reference_fe_t` (i.e. local FE space) to put atop each cell (see discussion above). The update of the `set_id` in the local cells is carried out within the search algorithm (Sect. 3.2), whereas the update in the off-processor ghost cells reuses an existing procedure in `triangulation_t` that invokes a nearest-neighbour communication. Another readily available method of `triangulation_t` takes charge of migrating the `set_id` values when the mesh is redistributed; see Sect. 3.1.

Apart from the special set up of `fe_space_t` described in the previous paragraph, `FEMPAR-AM` also requires to customize `fe_space_t` (as-is in `FEMPAR`) to support growing domains. In particular, one needs to inject DOF values of any field (e.g. temperature in thermal AM) from $\mathcal{T}_{h,j-1}$ into $\mathcal{T}_{h,j}$ and assign initial DOF values to *activated* cells. For this purpose, `FEMPAR-AM` implements `growing_fe_space_t`, a data type extending the standard `fe_space_t`, that provides an special method, referred to as `increment()`, performing this operation. The implementation of this procedure depends on how data

structures in charge of handling DOFs and DOF values are laid out and, more importantly, on whether each processor stores DOF values only in its local subdomain portion or also at the ghost cells layer. In the first case, an extra nearest-neighbour communication is needed to update DOF values at nodes sitting on a subdomain interprocessor interface.

The following paragraphs introduce the main software abstractions exclusive to `FEMPAR-AM`. They are in charge of supporting the update of the computational mesh, tracking the growth of the domain, and driving the main top-level AM process simulation loop. The software subsystem is formed by (1) `activation`, a customizable object enclosing data structures and methods necessary to find, at each time step, the subset of *activated* cells K_{acd}, (2) `cli_laser_path`, an AM-specific object to handle the geometrical information of the laser path, and (3) `discrete_path`, also AM-specific, to generate the space-time discretization of the laser path. To clarify their structure, contents and relationships, an UML class diagram is constructed in Fig. 6.

`activation` contains the `search_volume` abstract object, a placeholder for the geometrical description of the region to be activated during the time increment. Being a placeholder means that the instance is designed to allocate the minimum required memory space to hold a *single* search volume at a time. As the definition of the activated region depends on the application at hand, the client must specialize the behaviour of the object to its needs. In particular, extensions of this data type must have all the member variables and implement the methods needed to compute and update the dimensions and vertex coordinates defining the search cuboid at each time increment. In our context, `heat_affected_volume` is the AM-tailored search volume extended object. `activation` implements two public methods to satisfy the user requirements in the update of the computational mesh: (1) `update_search_volume(subsegment)` and (2) `overlaps_search_volume(cell)`. The former fills the search volume coordinates for a given time increment and the latter is used to test for collision between the search cuboid and any cell of the triangulation, using the method described in Sect. 3.2. Alg. 1 implements the update and increment of the computational domain, i.e. the $\mathcal{T}_{h,j-1}$ into $\mathcal{T}_{h,j}$ transformation, described in Sect. 3.2 and Fig. 3. For simplicity all steps regarding the treatment of the FE space and global FE functions are omitted, but they must be projected and redistributed. As observed, methods (1) and (2), invoked at Lines 1 and 7, fulfill important steps of the procedure.

Among the application-specific objects, `cli_laser_path` takes charge of managing the geometrical information of the laser path. The data comes in the same Common Layer Interface (CLI) ASCII file sent to the numerical control of the machine. CLI [69] is a common universal format for the input of geometry data to model fabrication systems based on layer manufacturing technologies. In the context of AM, a CLI file describes the movement of the laser in the plane of each layer with a complex sequence of *polylines*, to define the (smooth) boundary of the component, and *hatch* rectiliniar patterns for the inner structures (see [69] for several examples). `cli_laser_path` holds a parser for this file format (`parse(file_path)`) and accommodates its hierarchical structure in the following way: First, it aggregates an array of `layer` entities. For each *layer* in the CLI file, a `layer` instance is created and the layer height is stored. Likewise, for each *polyline* or *hatch* associated to the layer, an instance of `polyline` or `hatch` is created and filled with their plane coordinates. The class is supplemented with several query methods (e.g. `get_layer(layer_id)`, `get_hatch(hatch_id)`, `get_point(point_id)`), such that the user can navigate through the laser path subentities and extract their point coordinates. Although not implemented, a polymorphic superclass `laser_path` of `cli_laser_path` may be introduced to consider other file formats. In this way, new file formats can be accommodated with new child extensions of `laser_path`; at the moment, `FEMPAR-AM` can only support CLI files.

Closely associated to `cli_laser_path` is `discrete_path`, an entity that generates the space-time discretization of the laser path. Given that the printing process is tightly related to the movement of the laser, it is more natural to discretize the laser path with a *step length* Δx, instead of a time step. `discrete_path` takes the user-prescribed step length and the current *polyline* or *hatch* segment and divides it into subsegments of Δx size with the method `discretize(polyline/hatch)`. To support the time integration, `discrete_path` also computes the time increment associated to each subsegment as

$$\Delta t = \Delta \text{x} / \text{V}_{\text{scanning}}, \tag{8}$$

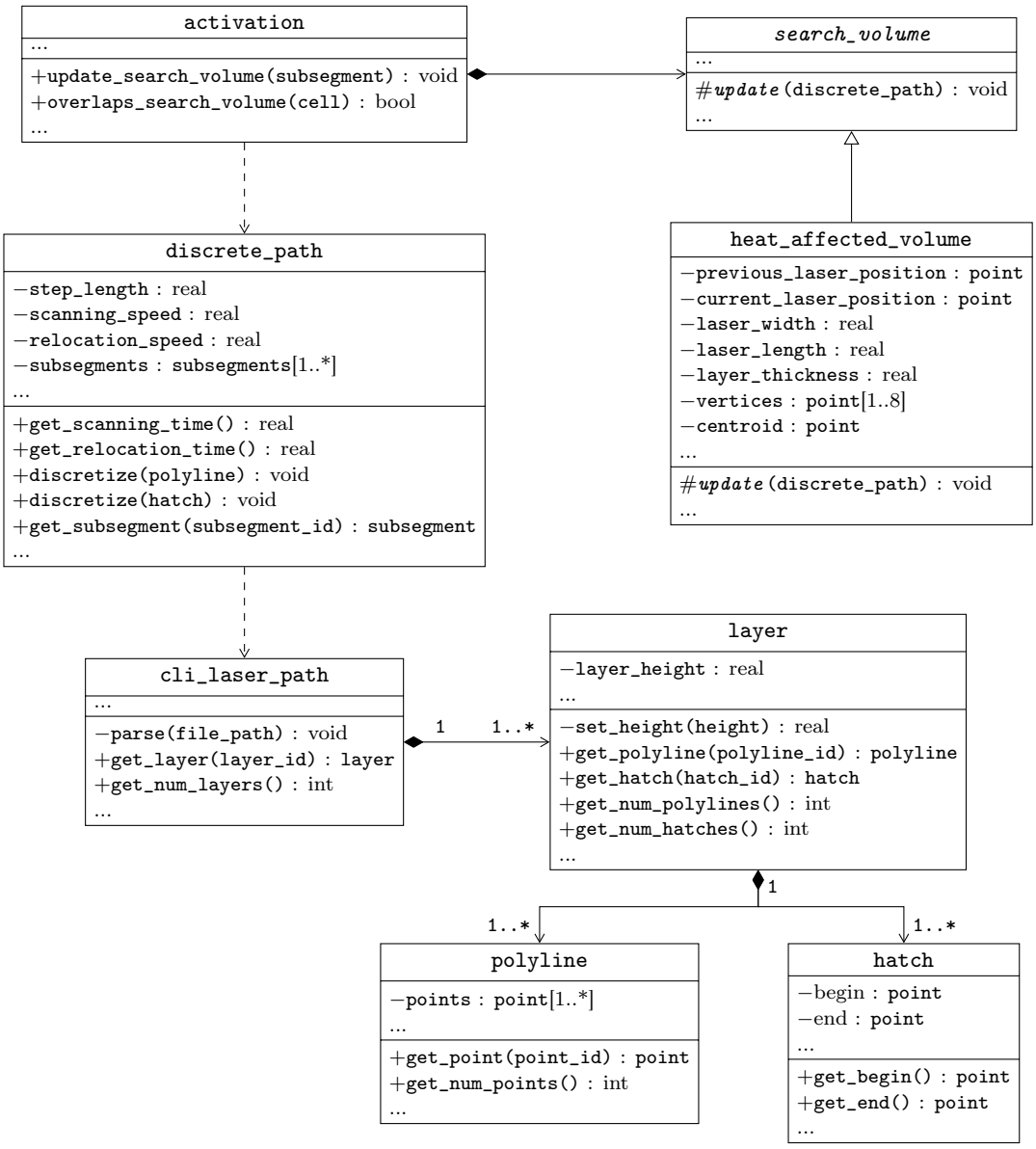


FIGURE 6. UML class diagram of the software subsystem that FEMPAR-AM uses to support the update of the computational mesh in Alg. 1 and to drive the AM process simulation, tracking the laser path, in Alg. 2. point is a simple class encapsulating three real-valued coordinates, whereas subsegment encapsulates two point instances.

where $V_{scanning}$ is the scanning speed. At each time step, discrete_path feeds activation with the current subsegment via update_search_volume(subsegment). After activation generates the current search volume and drives the mesh transformation, see Alg. 1, standard FE simulation steps follow until solving the linear system modelling the printing during the current time increment. The sequence is repeated until all subsegments have been simulated. Following this, the system is allowed to cool down, while the laser moves to the begin point of the next entity in the laser path or a new layer is spread. For the cooling step, discrete_path also computes recoat and relocation time as the

Algorithm 1: `update_and_increment_computational_domain(subsegment)` (see also Fig. 3)

```
activation.update_search_volume(subsegment)          /* fill search volume coordinates for current subsegment */
found_cell_for_refinement ← true
while found_cell_for_refinement do  /* repeat until all max level cells intersecting the search volume found */
    found_cell_for_refinement ← false
    for cell ∈ triangulation do
        if cell.is_local() then                                   /* local = owned by the current MPI task */
            if activation.overlaps_search_volume(cell) then  /* cell intersects the search volume (see Sect. 3.2) */
                if cell.get_level() < max_level_of_refinement then
                    cell.set_for_refinement()
                    found_cell_for_refinement ← true
                else
                    cell.set_for_do_nothing()
                end if
            else                                              /* cell does not intersect the search volume */
                if cell.get_level() > min_level_of_refinement then
                    cell.set_for_coarsening()
                else
                    cell.set_for_do_nothing()
                end if
            end if
        end if
    end for
    triangulation.refine_and_coarsen()
    ...                                  /* FE space and global FE functions projection/interpolation apply here */
    set_partition_weights()                                   /* construct weight function as in Eq. (1) */
    triangulation.redistribute()
    ...                                        /* FE space and global FE functions redistribution apply here */
end while
mark_activated_cells()
growing_fe_space.increment()      /* inject DOF values of T_{h,j-1} into T_{h,j} and initialize DOF values in K_acd */
```

quotient of the distance between the end point of the current *polyline* or *hatch* and the begin point of the next one divided by the relocation speed $V_{relocation}$.

A relevant feature of the design is that `discrete_path` is another placeholder container object, i.e. it only stores the discretization for the current *polyline* or *hatch* being printed and it is updated while looping over the laser path entities. Using the `cli_laser_path` recursive construction and the design of `discrete_path`, this loop, which is the one at the top level of `FEMPAR-AM` simulations, can be written in a compact form that reflects the actual printing process, as shown in Alg. 2 and Alg. 3. Note that the cost of the operations involved in filling and discretizing the laser path is negligible w.r.t. other stages of the simulation that concentrate the bulk of the computational cost (e.g. linear solver). Given this, and the fact that each processor must know the global search cuboid coordinates (see Sect. 3.2), data generated by `cli_laser_path` and `discrete_path` is not distributed, it is replicated in each processor to avoid extra communications.

Other minor thermal AM customizations that supplement the design are `heat_input`, a polymorphic and extensible entity with a suite of heat source term descriptions (e.g. uniform, surface Gauss or Goldak double ellipsoidal); `property_table`, an auxiliary object to allow the client to linearly interpolate properties that are known in tabular format, in order to evaluate temperature-dependent properties; and `heat_transfer_discrete_integration_t`, a subclass of `discrete_integration_t` [8, Sect. 11.2], in charge of evaluating the entries of the matrices and vectors defining the linear system of Eq. (6).

Algorithm 2: Top-level simulation loop of `FEMPAR-AM`

```
for layer ∈ laser_path do
    for polyline ∈ layer do
        discrete_path.discretize(polyline)          /* divide polyline into user-prescribed Δx-sized subsegments */
        simulate_printing_process(discrete_path)                                          /* see Alg. 3 */
    end for
    for hatch ∈ layer do
        discrete_path.discretize(hatch)             /* divide hatch into user-prescribed Δx-sized subsegments */
        simulate_printing_process(discrete_path)                                          /* see Alg. 3 */
    end for
end for
```

Algorithm 3: `simulate_printing_process(discrete_path)`

```
for subsegment ∈ discrete_path do
    update_and_increment_computational_domain(subsegment)                                 /* see Alg. 1 */
    ...   /* for simplicity, remaining simulation steps until linear system solution are omitted, but follow here */
end for
relocation_time ← discrete_path.get_relocation_time()
simulate_cooling(relocation_time)      /* solve Eq. (6) without heat input (laser relocation or layer recoat) */
```

6. Numerical experiments and discussion

6.1. **Verification of the thermal FE model.** First, the thermal FE model presented in Sect. 4 is verified against a 3D benchmark present in the literature [30, 46] that considers a moving single-ellipsoidal heat source on a semi-infinite solid with null fluxes at the free surface.

Assuming an Eulerian frame of reference (x, y, z), consider a heat source located initially at $z = 0$ that travels at constant velocity v along the x-axis on top of the semi-infinite solid defined by $z \leq 0$. The heat source distribution, derived from Goldak's double-ellipsoidal model [33], is defined by

$$q(x,y,z,t) = \frac{6\sqrt{3}}{\pi\sqrt{\pi}}\frac{Q}{abc}\exp\left[-3\left(\frac{(x-vt)^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\right)\right], \tag{9}$$

where Q is the (effective) rate of energy supplied to the system, v is the velocity of the laser and a, b, c are the main dimensions of the ellipsoid, as shown in Fig. 7(a).

The problem at hand is linear and admits a semi-analytical solution using Green's functions [16, 23] given by

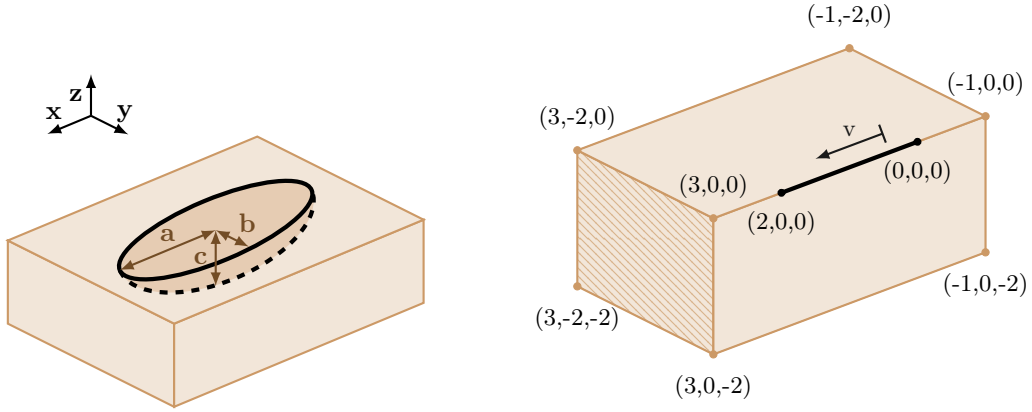
$$u(x,y,z,t) = u_0 + \frac{6\sqrt{3}}{\pi\sqrt{\pi}}\frac{\alpha Q}{k}\int_0^t \frac{\exp\left[-3\left(\frac{(x-vt)^2}{a^2+12\alpha(t-\tau)}+\frac{y^2}{b^2+12\alpha(t-\tau)}+\frac{z^2}{c^2+12\alpha(t-\tau)}\right)\right]}{\sqrt{(a^2+12\alpha(t-\tau))(b^2+12\alpha(t-\tau))(c^2+12\alpha(t-\tau))}}\mathrm{d}\tau, \tag{10}$$

with u_0 the initial temperature and $\alpha = k/\rho c$ the thermal diffusivity.

Using the symmetry of the problem, the numerical simulation considers a cuboid with coordinates given in Fig. 7(b). The path of the laser follows a segment centred along the edge of the cuboid that sits on the x-axis. Null fluxes apply at the top surface and the lateral face of symmetry, whereas Dirichlet boundary conditions apply at the remaining contour surfaces to account for the semi-infinite solid.

An h-adaptive linear FE mesh is employed, where the smallest size is prescribed around the welding path. Starting with the initial mesh shown in Fig. 8(a) and assigning $u_0 = 20$, $Q = 50$, $v = 1$, $\alpha = 0.1$, $k = 1$ and $(a, b, c) = (0.3, 0.15, 0.25)$, a convergence test is carried out.

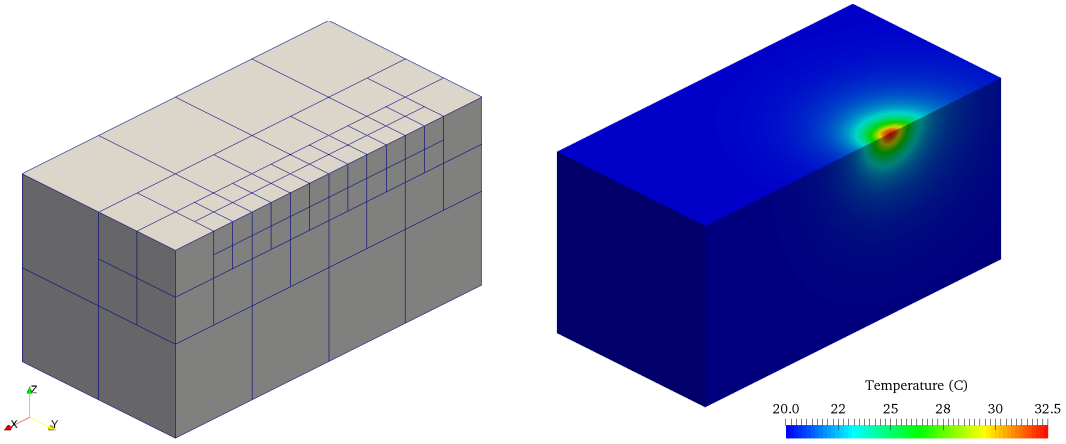
For a parabolic heat equation with sufficiently smooth solution, the error in $L^2([0,T]; H_0^1(\Omega))$ with a Backward Euler time integration scheme is proportional to $(h^p + \Delta t)$ (see Theorem 6.29 in [29]), where p is the order of the FE. Since $p = 1$ in this experiment, the time discretization should be refined at the same rate of the space discretization.



(A) Single-ellipsoidal heat source model.

(B) Domain of analysis, boundary conditions and welding path. Homogeneous Neumann boundary conditions apply at the top surface and the lateral face of symmetry. Dirichlet boundary conditions apply elsewhere.

FIGURE 7. 3D semi-analytical benchmark problem. Heat source distribution, geometry and boundary conditions.



(A) Initial adapted octree mesh.

(B) Temperature contour plot at $t = 0.5$ (3^{rd} refinement step).

FIGURE 8. 3D semi-analytical benchmark problem. Initial mesh and contour plot.

Taking this into consideration with an initial time step of $\Delta t = 0.008$, Fig. 9 shows that the numerical error in $L^2([0,T];H_0^1(\Omega))$ of the 3D semi-analytical benchmark decreases at the same rate as the theoretical one. This indicates a correct implementation of the thermal FE model.

6.2. **Strong-scaling analysis.** Next, the focus is turned to analysing the performance of `FEMPAR-AM` with a strong-scaling analysis[1]. The model problem for the subsequent experiments is designed to be

[1]Strong scalability is the ability of a parallel system (i.e. algorithm, software and parallel computer) to efficiently exploit increasing computational resources (CPU cores, memory, etc.) in the solution of a fixed-size problem. An *ideally* strongly scalable code decreases CPU time exactly as $1/P$, where P is the number of processors being used. In other words, if the system solves a size N problem in time t with a single processor, then it is able to solve the same problem in time t/P with P processors.

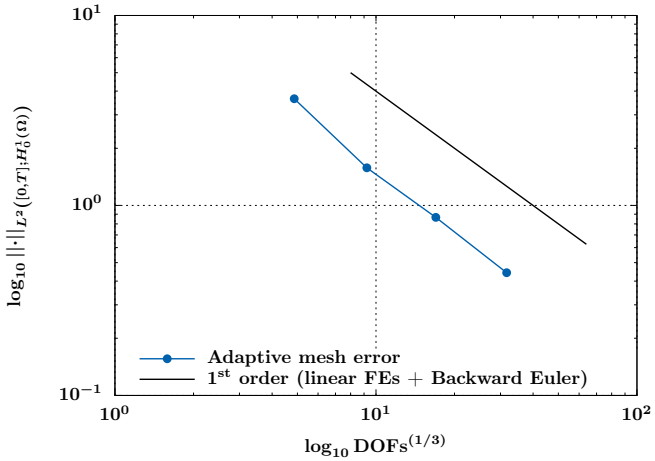


FIGURE 9. 3D semi-analytical benchmark problem. Convergence test.

geometrically simple, but with a computational load comparable to a real scenario of an industrial application.

According to this, the object of simulation is now the printing of 48 layers of 31.25 [μm] on top of a 32x32x16 [mm] prism. After printing the 48 layers, the prism has dimensions of 32x32x17.5 [mm], as shown in Fig. 10(a).

Concerning the process parameters, the power of the laser is set to 400 [W], the volumetric deposition rate during scanning is $d_\mathrm{p} = 10.0$ [mm^3/s] and the time allowed for lowering the platform, recoating and layer relocation between layers is $t_\mathrm{r} = 10.0$ [s].

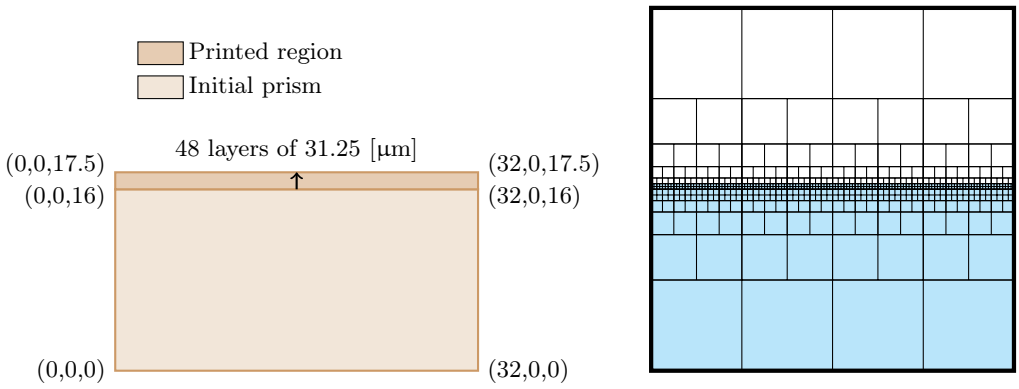
Apart from that, the material chosen is the Ti6Al4V Titanium alloy. The temperature dependent density, specific heat and thermal conductivity are obtained from a handbook and plotted in [21].

A constant heat convection boundary condition applies on the boundary of the cube with $h_\mathrm{out} = 50$ [W/m^{2}K] and $u_\mathrm{out} = 35$ [°C]. The initial temperature of both the prism and each new layer is $u_0 = 90$ [°C].

The root octant of the octree mesh is defined to cover a 32x32x32 [mm] cube region. The octree is transformed during the simulation to model the layer-by-layer deposition process, by prescribing a maximum refinement level of 11, i.e. assigning a mesh size of $h = 32,000/2^{11} = 15.625$ [μm], to the elements inside the layer that is currently being printed. A mesh gradation is then established according to the distribution of thermal gradients (highest at the printing region, lowest at the bottom of the cuboid). The mesh size in the (x, y) plane is fixed, whereas it decreases in both z-directions, until reaching a minimum level of refinement of 4 at the top and bottom of the prism. The computational domain is then defined by the initial prism and the layers that have been printed up to the current time. As seen in Fig. 10(b), most elements end up concentrating around the current layer, due to the coarsening induced by the 2:1 balance, but this is also the region with the highest temperature variations.

This refinement strategy leads to a simulation workflow that, for each layer, comprises the following steps:

(1) **Remeshing:** The previous mesh is refined and coarsened to accommodate the current layer.
(2) **Redistribution:** The new mesh is partitioned and redistributed among all processors to maintain load balance.
(3) **Activation:** New DOFs are distributed and initialized over the cells within the current layer.
(4) **Printing:** The problem is solved for the printing step. This step consists in the application of the heat needed to fuse the powder of the current layer and the time increment is calculated as $\Delta t = t_\mathrm{p} = \mathrm{V}_\mathrm{layer}/d_\mathrm{p}$ [s].



(A) Setting of the example. The initial prism and the printed region are plotted in light and dark brown.

(B) Illustration of the refinement strategy for maximum and minimum refinement levels of 8 and 3, resp. Computational domain in light cyan.

FIGURE 10. Strong-scaling problem set up. Plane XZ view of the setting and the mesh.

(5) **Cooling:** The problem is solved for the cooling step, accounting for lowering of building platform, recoat time and laser relocation with a time increment of $\Delta t = t_{\mathrm{r}}$ [s]. During the cooling step, the laser is off and the prism is allowed to cool down.

According to this workflow, the simulation of each layer is carried out in two time steps, printing and cooling, so the total number of time steps is $48 \cdot 2 = 96$. However, with the exception of the first layer, each new layer is meshed with a single refine and coarsen step. Hence, the simulation has about half as many AMR events as time steps. Besides, the linear system in Eq. (6), arising at each time step, is solved with the Jacobi-Preconditioned Conjugate Gradient (PCG) method with unit relaxation parameter.

The numerical experiments for this example were run at the Marenostrum-IV [54] (MN-IV) supercomputer, hosted by the Barcelona Supercomputing Center (BSC). It is equipped with 3,456 compute nodes connected together with the Intel OPA HPC network. Each node has 2x Intel Xeon Platinum 8160 multi-core CPUs, with 24 cores each (i.e. 48 cores per node) and 96 GBytes of RAM.

Apart from that, `FEMPAR-AM` was compiled with Intel Fortran 18.0.1 using system recommended optimization flags and linked against the Intel MPI Library (v2018.1.163) for message-passing and the BLAS/LAPACK library for optimized dense linear algebra kernels. All floating-point operations were performed in IEEE double precision.

The parallel framework is set up such that each subdomain is associated to a different MPI task, with a one-to-one mapping among MPI tasks and CPU cores. Regarding dynamic load balancing, the partition weights are set to $w_a = 10$ for active cells and $w_i = 1$ for inactive cells. Using linear FEs, the average total number of cells N_{cells} and global DOFs N_{dofs} (excluding hanging DOFs) across all time steps are 12,585,216 and 10,273,920. Note that, if a fixed uniform mesh was used, specifying the maximum refinement level of 11 all over the cube, the number of cells would be $(2^{11})^3 = 8.59 \cdot 10^9$ and the number of DOFs would grow from $(2^{11}+1)^2(2^{11}/2+1) = 4.30 \cdot 10^9$, initially, to $(2^{11}+1)^3 = 8.60 \cdot 10^9$, at the end of the simulation. Hence, it is readily exposed how h-adaptivity drastically reduces (almost by three orders of magnitude) the size of the problem and the required computational resources, while preserving accuracy around the growing printing region.

Fig. 11 and Tab. 1 report speed-up and total simulation wall time [s] of `FEMPAR-AM` using the Jacobi-PCG method, as the number of subdomains is increased. As observed, `FEMPAR-AM` scales up to 6,144 fine tasks with a peak speed-up of 19.2. Above 6,144 cores, time-to-solution increases due to parallelism related overheads (e.g. interprocessor communication); more computationally intensive simulations (i.e. larger loads per processor) would be required to exploit additional computational resources efficiently. As observed, the total wall time reduces with the number of subdomains to

approximately two minutes. This means that it takes 2.5 seconds in average to simulate the printing and cooling of a single layer (in two time steps). However, at larger scales of simulation and/or different problem physics, *weakly scalable* [2] methods, such as AMG or BDDC, may have superior performance.

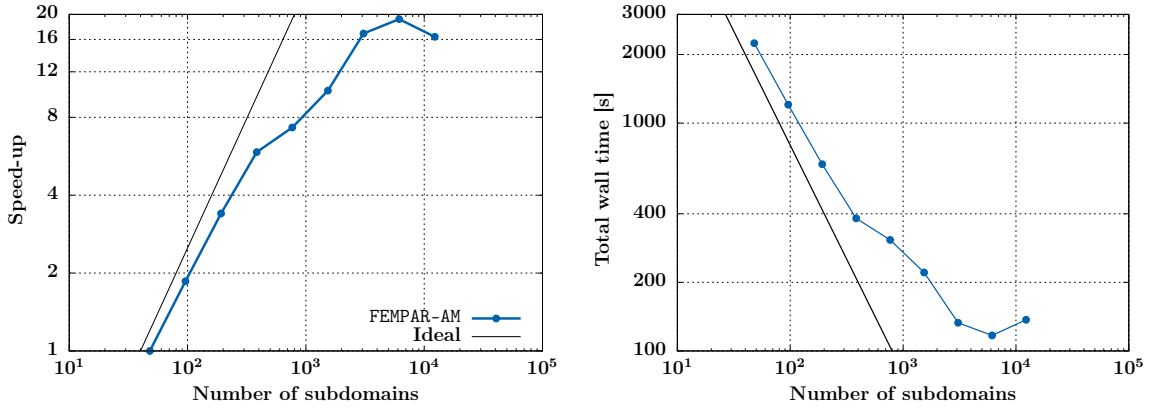


FIGURE 11. Strong-scaling example: Results of `FEMPAR-AM` for $(w_a, w_i) = (10, 1)$. Maximum speed-up of 19.2 and minimum wall time of 117 [s] is attained at 6,144 processors.

Another point of interest is to analyse the fraction of total wall time spent in different phases of the simulation and their scalability, shown in Fig. 12. As observed, the assembly phase dominates at low number of tasks, followed by the triangulation one. However, while assembly is the most scalable simulation phase, the triangulation is the least one. That is why the latter gradually dominates with increasing number of tasks and also leads the degradation of parallel efficiency. It is interesting to see that the solver phase is not relevant, with the exception of a (reproducible) spike in computation time at 786 and 1536 tasks. While the low-importance is caused by solving a relatively simple problem that can be efficiently preconditioned with the Jacobi method (the average number of iterations is merely 68), the abnormal deviation could be explained by the irregularity of the partition, although the authors were not able to find clear correlation. Another particularity related to the construction of the problem is that, when following a z-ordering, active and inactive cells are generally mixed. Hence, a standard $(w_a, w_i) = (1, 1)$ partition more or less equally distributes the active cells. It follows that there is a rather low sensitivity to the partition weights.

Further insights are drawn by studying the local distribution of cells and DOFs in space (among processors) and in time (among layers) up to 3,072 processors. As the number of cells and DOFs vary for each layer, all geometrical quantities are studied in terms of the mean μ and the coefficient of variation c_v, also known as relative standard deviation, which is the ratio of the standard deviation σ to the mean μ. c_v measures the extent of variability in relation to μ. Thus, it can be used to compare the variability among different quantities.

Given a quantity $x(t, p)$ that can depend on the time step and processor, the time average at every processor is represented as $\mu^t(x)$, the average among processors at every time step as $\mu^p(x)$ and the mean value across processors and time steps as $\mu(x)$. In what follows, the magnitude of x is studied with $\mu(x)$, whereas dispersion is analysed with the coefficient of variation among processors,

[2]Weak scalability is the ability of a parallel system to efficiently exploit increasing computational resources in the solution of a problem *with fixed local size per processor*. An *ideally* weakly scalable code does not vary the time-to-solution with the number of processors and fixed local size per processor. In other words, if the system solves a problem in time t with a given amount of processors, then it is able to solve also in time t an X times larger problem with X times the number of processors. The Jacobi method is not weakly scalable because the number of iterations grows with the global problem size.

P	**Total wall time [s]**	$\mathbf{S_P} = \frac{t_{48}}{t_P}$	$\mathbf{E_P} = \frac{S_P}{P/48}$	$\bar{\mathbf{n}}^{\mathbf{local}}_{\mathbf{dofs}}$	$\bar{\mathbf{n}}^{\mathbf{iters}}$
48	2,244	1.00	1.00	222,355	
96	1,205	1.86	0.93	111,812	
192	660	3.40	0.85	56,390	
384	382	5.87	0.73	28,533	
768	307	7.31	0.46	14,508	68
1,536	221	10.15	0.32	7,418	
3,072	133	16.87	0.26	3,827	
6,144	117	19.18	0.15	1,993	
12,288	137	16.38	0.06	1,052	

TABLE 1. Strong-scaling analysis results of `FEMPAR-AM` for $(w_a, w_i) = (10, 1)$. Total wall time accounts for the computational time of all simulation stages. S_P is the speed-up, E_P is the parallel efficiency, $\bar{n}^{\text{local}}_{\text{dofs}}$ is the average size of the local fine problem across processors and time steps and $\bar{n}^{\text{iters}}$ is the average number of iterations of the Jacobi-PCG solver across time steps.

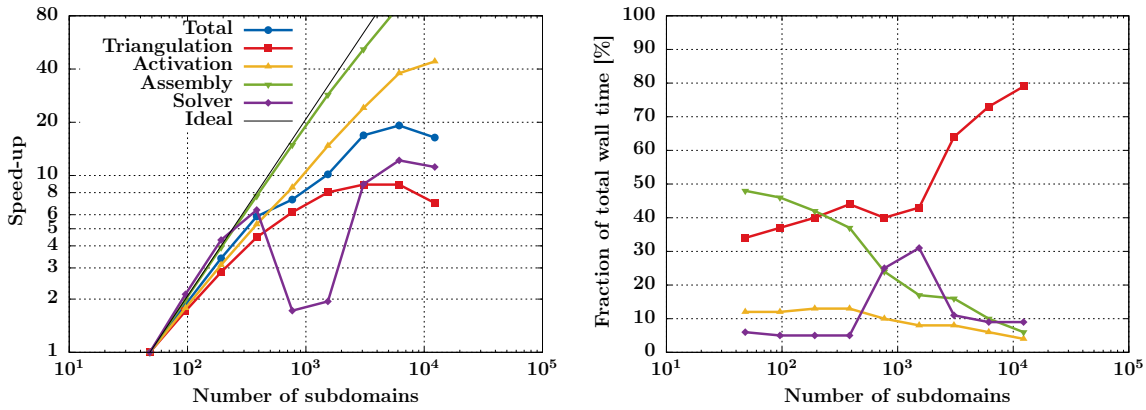


FIGURE 12. Strong-scaling example: Results of `FEMPAR-AM` for $(w_a, w_i) = (10, 1)$ per simulation phases. The *triangulation* phase accounts for the remesh and redistribute steps of the simulation workflow, including projections and redistributions of the FE solution, whenever the mesh is transformed or redistributed. The *activation* phase accounts for the search of activated cells and the generation of the FE space with new DOFs assigned within the current layer. The *assembly* phase consists of local integration of the weak form and construction of the global linear system, during printing and cooling steps. Finally, the *solver* phase includes the solution of the linear system, also during printing and cooling steps. Although the assembly phase is initially dominant, computational time and scalability are dominated by the triangulation phase.

i.e. $c_v^p(x) = \sigma^p(x)/\mu^p(x)$. This statistic informs about possible computational load unbalances, due

to an uneven distribution of x among processors. As $c_v^p(x)$ depends on the time step, for the sake of simplicity, the average across time steps $\mu^t(c_v^p(x))$ is reported instead.

Tab. 2 gathers the local distribution of cells and degrees of freedom (excluding the hanging ones). The values of $\mu^t(c_v^p(x))$ show that the local number of cells is slightly unbalanced, but the local weighted number of cells, i.e. the sum of the cell weights at each processor for $(w_a, w_i) = (10, 1)$, is perfectly balanced. Apart from that, Fig. 13 shows that the number of cells oscillates with the height of the layer, but it does not grow in time. This behaviour propagates to other quantities such as the number of active cells or DOFs and it is caused by the 2:1 balance: The cells concentrate at the current layer and immediately below. Even if the domain grows in time, the number of cells away from the layer is much smaller than the number of those close or at the layer.

	$\mathbf{n_{cells}}$		$\mathbf{n_{cells}^{weighted}}$		$\mathbf{n_{cells}^{active}}$		$\mathbf{n_{dofs}}$	
P	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$
48	262.2k	0.61	2,228k	0.00	218.5k	0.08	222.4k	0.26
96	131.0k	0.75	1114k	0.00	109.2k	0.10	111.8k	0.37
192	65.5k	1.02	557k	0.01	54.6k	0.14	56.4k	0.45
384	32.8k	1.44	279k	0.01	27.3k	0.20	28.5k	0.63
768	16.4k	2.28	139k	0.02	13,7k	0.31	14.5k	0.73
1,536	8.2k	3.46	69.6k	0.05	6,8k	0.48	7.4k	1.10
3,072	4.1k	5.46	35.8k	0.09	3,4k	0.76	3.8k	1.39

TABLE 2. Strong-scaling example. `FEMPAR-AM` for $(w_a, w_i) = (10, 1)$. Local distribution of cells and degrees of freedom (excluding hanging). $\mu^t(c_v^p)$ expressed in %.

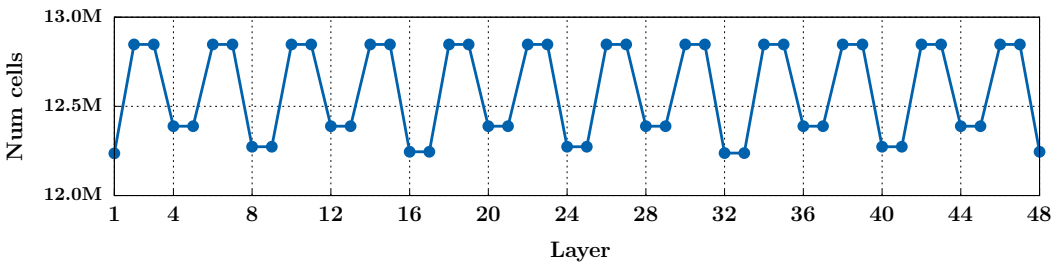


FIGURE 13. Evolution of global number of cells with the height of the layer.

It is also apparent that the pair $(w_a, w_i) = (10, 1)$ effectively equilibrates the number of active cells among processors. Hence, degrees of freedom are also evenly distributed, though with a slightly higher dispersion. This is especially beneficial for the integration and assembly phases, as they are implemented in `FEMPAR`, such that the bulk of the computational load is concentrated on the active cells set, and also the linear solver phase, as the size of the local systems depend on the number of DOFs that the processor owns. On the other hand, mesh generation, refinement, coarsening and redistribution phases suffer from an uneven distribution of total cells.

Concerning the local number of subdomain neighbours and interface DOFs, in Tab. 3, high variations in space expose the extreme irregularity of Z-curve partitions. Due to the refinement strategy in Fig. 10(b) and the Z-ordering, most subdomains are embedded in the current layer, while only few are made of bottom coarser cells. This explains why, as seen in the fourth column, listing the maximum

number of neighbours across processors and time steps, there can be subdomains touching all the remaining ones. Even if the partition becomes increasingly regular with P, the imbalance of neighbours and interface DOFs increases synchronization times in important operations of the Jacobi-PCG, such as the matrix-vector product. It could also be the main cause for the deviation observed in the solver times of Fig. 12.

	$\mathbf{n_{neighbours}}$			$\mathbf{n_{dofs}^{interface}}$	
P	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$	**max**	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$
48	14	47.9	48	7.1k	35.1
96	16	62.5	95	4.8k	34.4
192	17	79.4	191	3.3k	28.3
384	18	91.1	383	2.3k	25.8
768	19	103.6	767	1.6k	20.5
1,536	20	92.0	1,024	1.1k	20.7
3,072	21	80.1	1,048	0.8k	16.6

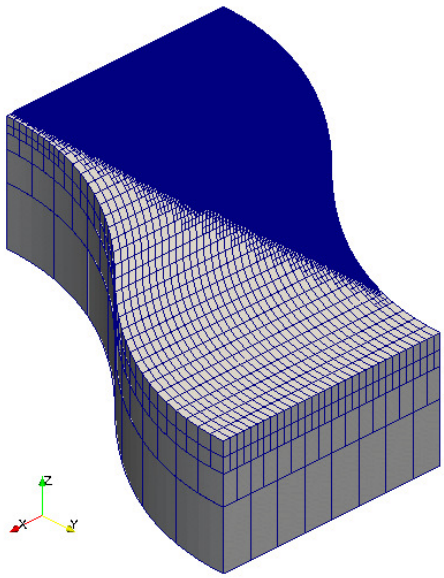
TABLE 3. Strong scaling example. Local distribution of subdomain neighbours and interface degrees of freedom. $\mu^t(c_v^p)$ and c_v expressed in %.

6.3. **Extruded wiggle.** If the previous example focused in performance, the one in this section aims to show the capabilities of the framework in a more realistic scenario. The setting now is the printing of a 3D curved geometry *following the actual scan path*, instead of a layer-averaging approach. In this way, the geometry is no longer rectangular, the temperature field is captured with higher detail and, more importantly, the parallel search algorithm introduced in Sect. 3.2 is more intensively stressed.

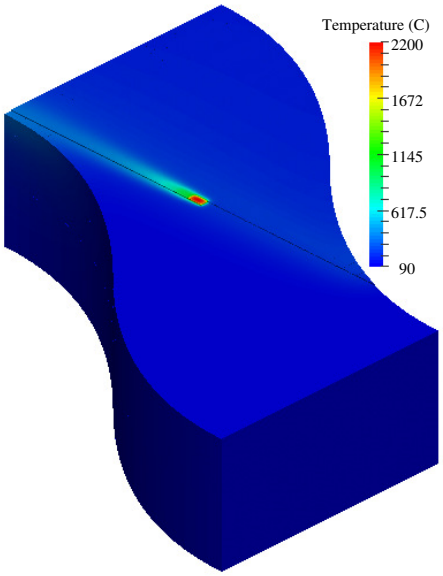
The simulation spans the build of eight 60 [µm] layers on top of a 7.68 [mm] height extruded wiggle. The geometry, represented in Fig. 14, is obtained in the following way: (1) a one-dimensional wiggle is defined by joining two parabolic functions, the distance between its edges is 30.72 [mm]; (2) extrusion along the x-axis yields a 15.36 [mm] thick plane wiggle; (3) extrusion perpendicular to the xy-plane completes the 3D shape.

One feature of the simulation is the use of *second order* lagrangian FEs both for geometry and continuous FE space approximations. In particular, given that the wiggle is parametrized by (piecewise) polynomials of order less or equal than two, an exact discretization can be easily defined. Although this choice clashes with the rectangularity assumption of the search algorithm in Sect. 3.2, if the HAV overlaps a refined enough region, the mesh cells can be regarded as quasi-rectangular. As a result, the meshing approach considers two different user-prescribed minimum levels of refinement: the (1) *absolute* one, a lower bound of the level of any cell in the mesh, and the (2) *search* one, i.e. the lowest level a cell below the layer being printed must have to ensure quasi-rectangularity. It is apparent that (1) is always lower or equal than (2).

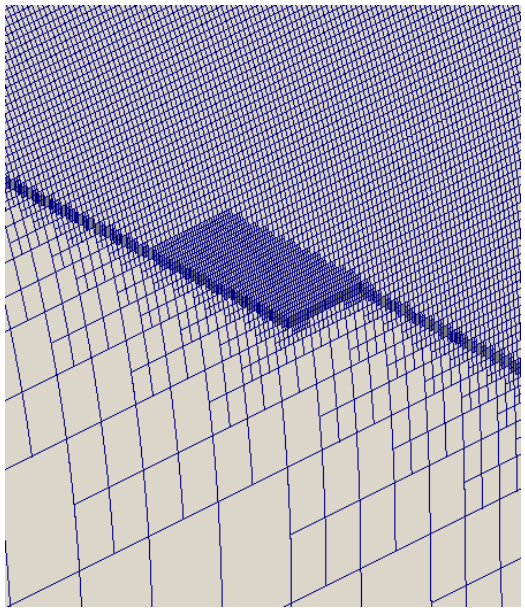
The octree mesh is constructed by working on two different configuration spaces; `p4est` generates and transforms the mesh in a *reference* unit cube, whereas `FEMPAR` maps the unit cube mesh coordinates into the *physical* wiggle. The root octant of the octree mesh is mapped into a 30.72 [mm] height wiggle. The absolute minimum level of refinement is set to three, enough to have an exact discrete geometry; the search one is set to five, after several trial and error experiments; and the maximum one at the HAV is ten, i.e. there are two cells along the thickness of the layer. This setting is apparent in Figs. 14(a) and 14(c). The initial computational domain has a height of 7.68 [mm], cells located above this threshold are inactive. After printing the eight layers, the active region reaches 8.16 [mm] height.



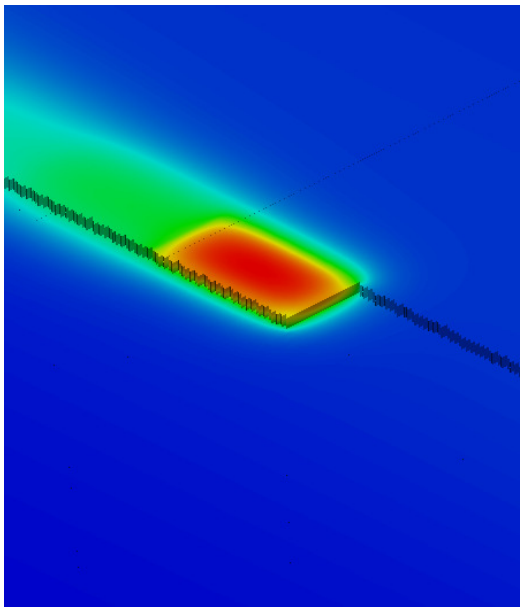
(A) The absolute minimum level of refinement at the bottom of the wiggle is three, while the search one below the current layer is at least five. A blue shaded area indicates the highly-refined cell concentration at the current layer.



(B) Contour plot of temperatures for the given time step. The relevant thermal features are localized around the HAV and its tail.



(C) Close-up of the mesh around the HAV. The search algorithm finds the cells inside the HAV on top of a quasi-rectangular region and assigns to them the maximum level of refinement (10).



(D) Close-up of the contour plot around the HAV. Away from the HAV, highly-refined layer cells are only necessary to capture the growing geometry; they do not increase the resolution.

FIGURE 14. Mesh and temperature contour plot of the 3D wiggle at an intermediate time step of the simulation of the first layer for $(w_a, w_i) = (10, 1)$.

Regarding the laser path, odd layers are built with nonoverlapping hatches along the x-axis, whereas even ones along the y-axis. Even though the scanning path is orthogonal, the mesh cells are skewed due to the cube-to-wiggle mapping. Therefore, during the simulation, the search algorithm generally tests the intersection of non-aligned cuboids. A uniform heat input of 200 [W] and heat absorption η of 0.5 is distributed in a 0.96x0.48x0.06 [mm] HAV. Note that the HAV has been scaled 1% extra in the xy-plane for safety, a practice that is advisable even for linear FEs to avoid failures of the search algorithm in situations that depend on arithmetic precision. The scanning speed is 100 [mm/s] and the relocation speed is 200 [mm/s]. This means that the time step during printing is $0.96/100 = 9.6 \cdot 10^{-3}$ [s].

The material is, once again, Ti6Al4V, and the initial and boundary conditions are also the same of the previous example, i.e. $u_0 = 90$ [°C] and convection on the boundary ($h_{\text{out}} = 50$ [W/m^{2}K] and $u_{\text{out}} = 35$ [°C]). An important simplification is that the powder is not simulated. Although modelling the powder bed has already been object of study by the authors [58], it is left out of this example, because it does not contribute to the main purpose, i.e. the study of the search algorithm.

The simulation workflow is the same as the one in Sect. 6.2, but the time discretization is set up such that, at each time step, 0.96 [mm] along the laser path are advanced, not a whole layer. Simulation steps (1) to (4), i.e. mesh transformation, activation and printing step, are carried out at every 0.96 [mm] step, while cooling (5) steps only apply between hatches and layers. In this strategy, the mesh explicitly tracks the laser path, but there are many more AMR events than time steps, because the next HAV is not accommodated in a single refine and coarsen step, but (at most) $11 - 5 = 6$ steps. Although not covered in this work, other simulation strategies could be analysed. For instance, one could mesh the whole layer as in Sect. 6.2, then simulate printing following the laser path point to point. Compared to the laser-tracking mesh approach, there would be many more time steps than mesh transformations, but the problem solved at each time step would also be bigger.

Numerical experiments were run with 96 MPI tasks one-to-one assigned to 96 cores, distributed in six computing nodes. Each node has 2x Intel Xeon E5-2670 multi-core CPUs, with 8 cores each and 64 GBytes of RAM. The computing nodes were located at the Acuario [1] computer cluster, hosted by the International Centre of Numerical Methods in Engineering (CIMNE) in Barcelona, Spain. Linear systems were solved with a nonrelaxed Jacobi-PCG method. Contour plots were generated in ParaView 5.1.0 [3] with second order visualization cells written in VTK format [65].

Simulation results at several time steps are gathered in Figs. 14 and 16 for $(w_a, w_i) = (10, 1)$. Average global problem size is 2.0M (nonhanging) DOFs and total number of time steps is 8,727. For 96 MPI tasks, the average number of local DOFs is 21.1k and $\mu^t(c_v^p(x)) = 3.3\%$, with $c_v^p(x) = \sigma^p(x)/\mu^p(x)$. In front of the strong-scaling example, dispersion of local DOFs is higher and the total number of DOFs notably oscillates in time (Fig. 15) due to the laser-tracking refinement strategy.

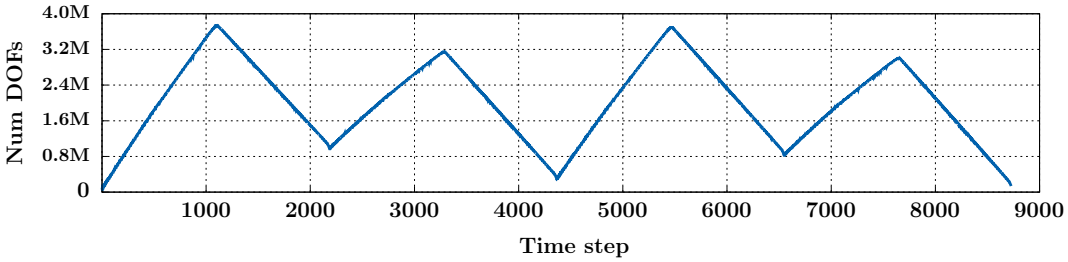
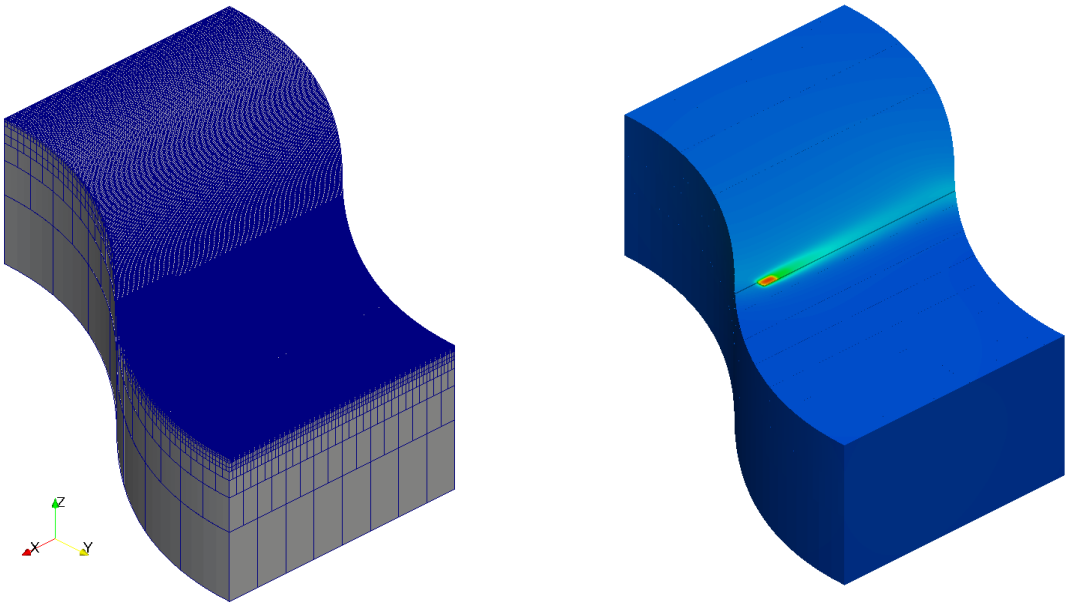
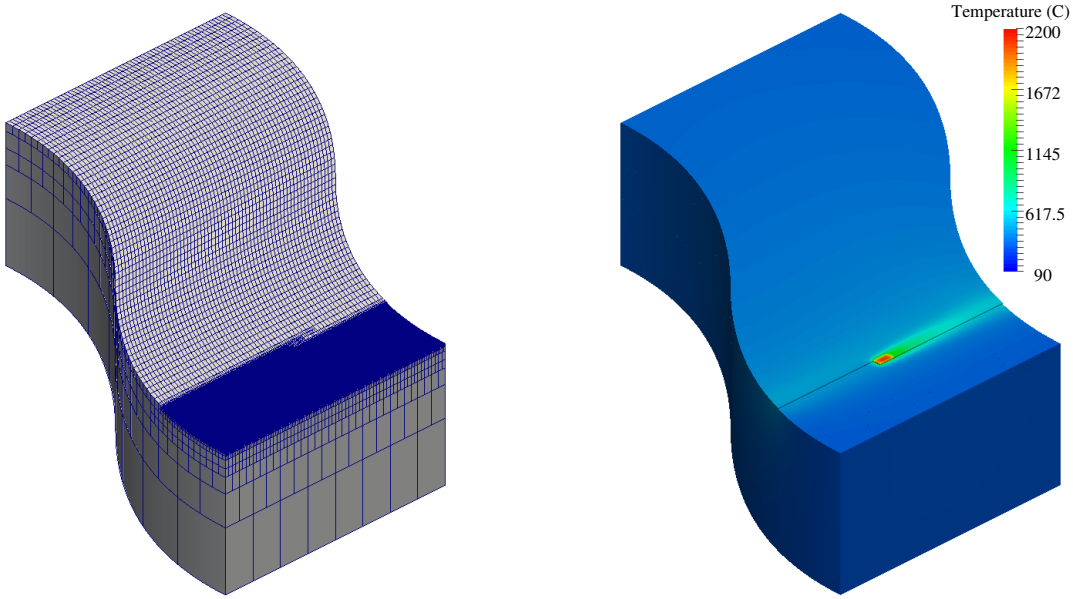


FIGURE 15. Total number of (nonhanging) dofs heavily oscillates with the laser-tracking refinement approach. Increasing/decreasing trend reverses at the start of a new layer.

The duration of the simulation is 13 [h]; each layer is printed in 97 [min] average, but the individual time varies significantly, due to the oscillation of total number of DOFs. On the other hand, average number of Jacobi-PCG iterations is 381. In spite of being numerous, mesh generation, adaptation and redistribution roughly amounts to 52% of the simulation time. Next phases by runtime are local



(A) As the mesh must be layer-conforming, many cells concentrate at the last printed layers, even though they do not contribute to capture better the thermal field.



(B) Homogeneous (same level) coarsening at the last layer indicates that there are no interior or boundary holes, i.e. cells that the search algorithm has failed to activate.

FIGURE 16. Mesh and temperature field at intermediate time steps of the sixth 16(a) and eighth 16(b) layers.

integration and assembly (22%) and linear solver (24%). An important outcome is that activation with the HST nonintersection test and initialization of new DOFs only occupies 3% of the simulation.

Moreover, mesh and contour plots (e.g. Fig. 16(b)) do not expose any holes during the filling of the layers. Therefore, the parallel search algorithm is both efficient and robust in this example.

Apart from that, there is a pronounced sensitivity to the partition weights. Indeed, Tab. 4 gathers distribution of DOFs and runtimes for several pairs of (w_a, w_i). A weighted partition decreases up to 34% the computation time with respect to a nonweighted one. Runtime decrease affects especially the assembly and solver phases, whose complexity depends on the number of active cells or DOFs, respectively. Reductions could be more significant for larger number of subdomains P, because the disequilibrium of local DOFs grows with P. Hence, partition weights not only dynamically equilibrate the DOFs among processors, they also have the potential to significantly shorten simulation times. Besides, in relation to the discussion at the end of Sect. 3.3, performance of pairs (w_a, w_i) satisfying $w_a \gg w_i$ is barely distinguished from $(w_a, w_i) = (1, 0)$, i.e. they appear capable of equilibrating the computational load, while precluding extreme imbalance of cells.

	$\mathbf{n_{dofs}}$		Runtime		Phase percentage [%]			
$(\mathbf{w_a}, \mathbf{w_i})$	$\boldsymbol{\mu}$	$\boldsymbol{\mu^t(c_v^p)}$ [%]	T [h]	$\mathrm{T/T}_{(1,1)}$ [%]	Triang.	Activation	Assembly	Solver
(1,1)	21.1k	25.6	19.7	100	48	3	26	23
(2,1)		15.4	15.2	77	51	2	23	24
(10,1)		3.3	13.0	66	52	3	22	24
(100,1)		3.0	13.0	66	52	2	23	23
(1,0)		2.4	13.0	66	51	2	23	24

TABLE 4. Sensitivity of DOF distribution and computation times to the partition weights. Simulation phases correspond to the same ones described in Fig. 12. Compared to the nonweighted case, partition weights equilibrate the DOF distribution and reduce up to 34% the total simulation runtime.

A final comment arises on two computational bottlenecks identified in this example that guide how the framework could be further improved, towards dealing with real industrial scenarios. The first and most obvious one is the layer-conforming mesh constraint that precludes coarsening at cells touching the active/inactive interface (e.g. Fig. 16(a)). Unfitted FE methods could likely get rid of this requirement and dramatically drop the mesh size. On the other hand, even if concurrency in space is efficiently exploited, the framework is still fully sequential in time. Very small time steps must be prescribed to follow the laser with high precision, leading to extremely long transient simulations. This motivates exploring adaptive space-time approximations and solvers [4, 31, 32]. The idea is to provide the solution at all time steps in one shot using the vast computational resources available nowadays.

7. CONCLUSIONS

This work has introduced a novel HPC numerical framework for growing geometries that has been applied to the thermal FE analysis of metal additive manufacturing by powder-bed fusion at the part scale. The abstract framework is constructed from three main blocks (1) hierarchical AMR with octree meshes, distributed across processors with space-filling curves; (2) the extension of the element-birth method to (1); to track the growth of the geometry with an embarrassingly parallel search algorithm, based on the hyperplane separation theorem; and (3) state-of-the-art parallel iterative linear solvers.

After implementation in `FEMPAR`, an advanced object-oriented FE code for large scale computational science and engineering, a strong-scaling analysis, of a problem with 10M unknowns, evaluated the performance of the code up to 12,288 CPU cores. Timing of simulation phases and statistical treatment of cell- and DOF-related quantities revealed uneven DOF distribution and partition irregularity as the main sources of load imbalance, thus increasing parallel overhead.

A second numerical example of 2.0M unknowns in a curved geometry examined the search algorithm, driving the update of the computational domain. The results not only verified efficiency and robustness of the search routine, but also showed that the mesh rectangularity assumption of the algorithm can be slightly relaxed. A further study exposed the potential of partition weights to tune dynamic load balance and bring down simulation times; the weight function has been defined to seek a compromise between equally distributing among processors the number of cells or DOFs.

Average simulation times *per time step* were cut down to 1.2 [s] and 5.4 [s] in the first and second examples. Even at the rate of 1 [s] per time step, fully resolved high-fidelity AM simulations, involving, e.g. other physics than heat transfer, stronger nonlinearities, historical variables, among others, can still take long hours in massively parallel systems. Higher efficiency and parallelism could be attained by resorting to (1) unfitted FE methods [10] to eliminate the mesh body- *and* layer-fitting requirement and (2) weakly scalable adaptive and nonlinear space-time solvers that also exploit concurrency in time.

In spite of the limitations, `FEMPAR-AM` is, to the authors knowledge, the first *fully* parallel and distributed FE framework for part-scale metal AM and the numerical experiments have shown the potential to efficiently address levels of complexity and accuracy unseen in the literature of metal AM. Apart from turning to other part-scale physics in metal or polymer AM, this HPC framework could also be useful for other growing-geometry problems, as long as the growth is modelled by adding new elements into the computational mesh.

8. Acknowledgements

Financial support from the EC - International Cooperation in Aeronautics with China (Horizon 2020) under the *EMUSIC* project (*Efficient Manufacturing for Aerospace Components USing Additive Manufacturing, Net Shape HIP and Investment Casting*), the EC - Factories of the Future (FoF) Programme under the *CA×Man* Project (*Computer Aided Technologies for Additive Manufacturing*) within *Horizon 2020* Framework Programme and the Spanish Government-MINECO-Proyectos de I+D (Excelencia)-DPI2017-85998-P-ADaMANT-Computational Framework for Additive Manufacturing of Titanium Alloy are gratefully acknowledged. The authors thankfully acknowledge the computer resources at Titani (Camins-UPC), Acuario (CIMNE), Marenostrum-IV (RES-ActivityIDs: FI-2018-1-0014, FI-2018-2-0009, FI-2018-3-0029 and FI-2019-1-0007) and the technical support provided by the Barcelona Supercomputing Center. E. Neiva gratefully acknowledges the support received from the Catalan Government through a FI fellowship (2018 FI-B1-00095; 2017 FI-B-00219). S. Badia gratefully acknowledges the support received from the Catalan Government through the ICREA Acadèmia Research Program. Financial support to CIMNE via the CERCA Programme of the Generalitat de Catalunya is also acknowledged.

References

[1] Acuario website. https://hpc.cimne.upc.edu/, (n.d.). Accessed: 2019-03-02.

[2] A. Anca, V. D. Fachinotti, G. Escobar-Palafox, and A. Cardona. Computational modelling of shaped metal deposition. *International Journal for Numerical Methods in Engineering*, 85(1): 84–106, 2011.

[3] U. Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

[4] S. Badia and M. Olm. Space-time balancing domain decomposition. *SIAM Journal on Scientific Computing*, 39(2):C194–C213, 2017.

[5] S. Badia and F. Verdugo. Robust and scalable domain decomposition solvers for unfitted finite element methods. *Journal of Computational and Applied Mathematics*, 2017.

[6] S. Badia, A. F. Martín, and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*, 36(2):C190–C218, 2014.

[7] S. Badia, A. F. Martín, E. Neiva, and F. Verdugo. A generic finite element framework on parallel tree-based adaptive meshes. *Submitted*, 2018.

[8] S. Badia, A. F. Martín, and J. Principe. `FEMPAR`: An object-oriented parallel finite element framework. *Arch. Comput. Methods Eng.*, 25(2):195–271, 2018.
[9] S. Badia, A. F. Martín, and J. Principe. `FEMPAR` Web page. `http://www.fempar.org`, 2018.
[10] S. Badia, F. Verdugo, and A. F. Martín. The aggregated unfitted finite element method for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 336:533–553, 2018.
[11] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Transactions on Mathematical Software (TOMS)*, 38(2):14, 2011.
[12] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
[13] G. Bugeda, M. Cervera, and G. Lombera. Numerical prediction of temperature and density distributions in selective laser sintering processes. *Rapid Prototyping Journal*, 5(1):21–26, 1999.
[14] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011. doi: 10.1137/100791634.
[15] C. Burstedde, J. Holke, and T. Isaac. On the number of face-connected components of morton-type space-filling curves. *Foundations of Computational Mathematics*, 2018. doi: 10.1007/s10208-018-9400-5.
[16] H. Carslaw and J. Jaeger. *Conduction of heat in solids*. Oxford Science Publications: Oxford, UK, 1959.
[17] M. Cervera, C. Agelet De Saracibar, and M. Chiumenti. Thermo-mechanical analysis of industrial solidification processes. *International Journal for Numerical Methods in Engineering*, 46(9):1575–1591, 1999.
[18] M. Chiumenti, C. A. de Saracibar, and M. Cervera. On the numerical modeling of the thermomechanical contact for metal casting analysis. *Journal of Heat Transfer*, 130(6):061301, 2008.
[19] M. Chiumenti, M. Cervera, N. Dialami, B. Wu, L. Jinwei, and C. Agelet de Saracibar. Numerical modeling of the electron beam welding and its experimental validation. *Finite Elements in Analysis and Design*, 121:118–133, 2016.
[20] M. Chiumenti, X. Lin, M. Cervera, W. Lei, Y. Zheng, and W. Huang. Numerical simulation and experimental calibration of additive manufacturing by blown powder technology. part i: thermal analysis. *Rapid Prototyping Journal*, 23(2):448–463, 2017.
[21] M. Chiumenti, E. Neiva, E. Salsi, M. Cervera, S. Badia, J. Moya, Z. Chen, C. Lee, and C. Davies. Numerical modelling and experimental validation in selective laser melting. *Additive Manufacturing*, 18(Supplement C):171 – 185, 2017. ISSN 2214-8604. doi: https://doi.org/10.1016/j.addma.2017.09.002.
[22] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang. A survey of parallelization techniques for multigrid solvers. In *Parallel processing for scientific computing*, pages 179–201. SIAM, 2006.
[23] K. D. Cole, J. V. Beck, A. Haji-Sheikh, and B. Litkouhi. *Heat conduction using Green's functions*. CRC Press, 2010.
[24] C. A. De Saracibar, M. Cervera, and M. Chiumenti. On the formulation of coupled thermoplastic problems with phase-change. *International journal of plasticity*, 15(1):1–34, 1999.
[25] E. R. Denlinger, V. Jagdale, G. V. Srinivasan, T. El-Wardany, and P. Michaleris. Thermal modeling of Inconel 718 processed with powder bed fusion and experimental validation using in situ measurements. *Additive Manufacturing*, 11:7–15, 2016. ISSN 2214-8604.
[26] E. R. Denlinger, M. Gouge, J. Irwin, and P. Michaleris. Thermomechanical model development and in situ experimental validation of the Laser Powder-Bed Fusion process. *Additive Manufacturing*, 16:73–80, 2017.
[27] A. J. Dunbar, E. R. Denlinger, M. F. Gouge, and P. Michaleris. Experimental validation of finite element modeling for laser powder bed fusion deformation. *Additive Manufacturing*, 12, Part A: 108–120, 2016.
[28] D. Eberly. Dynamic collision detection using oriented bounding boxes. *Geometric Tools, Inc*, 2002.

[29] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159. Springer Science & Business Media, 2013.

[30] V. D. Fachinotti, A. A. Anca, and A. Cardona. Analytical solutions of the thermal field induced by moving double-ellipsoidal and double-elliptical heat sources in a semi-infinite body. *International Journal for Numerical Methods in Biomedical Engineering*, 27(4):595–607, 2011.

[31] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[32] M. J. Gander. 50 years of time parallel time integration. In *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113. Springer, 2015.

[33] J. Goldak, A. Chakravarti, and M. Bibby. A new finite element model for welding heat sources. *Metallurgical transactions B*, 15(2):299–305, 1984.

[34] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[35] S. Gottschalk, D. Manocha, and M. C. Lin. *Collision queries using oriented bounding boxes*. PhD thesis, University of North Carolina at Chapel Hill, 2000.

[36] H. Haverkort and F. van Walderveen. Locality and bounding-box quality of two-dimensional space-filling curves. In *European Symposium on Algorithms*, pages 515–527. Springer, 2008.

[37] N. E. Hodge, R. M. Ferencz, and R. M. Vignes. Experimental comparison of residual stresses for a thermomechanical model for the simulation of selective laser melting. *Additive Manufacturing*, 12, Part B:159–168, 2016.

[38] J. Irwin and P. Michaleris. A Line Heat Input Model for Additive Manufacturing. *Journal of Manufacturing Science and Engineering*, 138(11):111004–111004–9, 2016.

[39] T. Isaac, C. Burstedde, L. C. Wilcox, and O. Ghattas. Recursive algorithms for distributed forests of octrees. *SIAM Journal on Scientific Computing*, 37(5):C497–C531, 2015.

[40] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. doi: 10.1137/S1064827595287997.

[41] G. Karypis and V. Kumar. Parallel Multilevel series k-Way Partitioning Scheme for Irregular Graphs. *SIAM Review*, 41(2):278–300, 1999. ISSN 0036-1445. doi: 10.1137/S0036144598334138.

[42] L. Kaufman and H. Bernstein. *Computer calculation of phase diagrams with special reference to refractory metals*. Academic Press New York, 1970. ISBN 012402050.

[43] T. Keller, G. Lindwall, S. Ghosh, L. Ma, B. M. Lane, F. Zhang, U. R. Kattner, E. A. Lass, J. C. Heigel, Y. Idell, et al. Application of finite element, phase-field, and calphad-based methods to additive manufacturing of ni-based superalloys. *Acta materialia*, 139:244–253, 2017.

[44] A. Kergaßner, J. Mergheim, and P. Steinmann. Modeling of additively manufactured materials using gradient-enhanced crystal plasticity. *Computers & Mathematics with Applications*, 2018.

[45] S. A. Khairallah, A. T. Anderson, A. Rubenchik, and W. E. King. Laser powder-bed fusion additive manufacturing: Physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones. *Acta Materialia*, 108:36–45, 2016.

[46] S. Kollmannsberger, A. Özcan, M. Carraturo, N. Zander, and E. Rank. A hierarchical computational model for moving thermal loads and phase changes with applications to selective laser melting. *Computers and Mathematics with Applications*, 75(5):1483 – 1497, 2018. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2017.11.014.

[47] S. Kolossov, E. Boillat, R. Glardon, P. Fischer, and M. Locher. 3d FE simulation for temperature evolution in the selective laser sintering process. *International Journal of Machine Tools and Manufacture*, 44(2–3):117–123, 2004.

[48] P. Küs and J. Šístek. Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver. *Advances in Engineering Software*, 110:34–54, 2017.

[49] Y. Lian, S. Lin, W. Yan, W. K. Liu, and G. J. Wagner. A parallelized three-dimensional cellular automaton model for grain growth during additive manufacturing. *Computational Mechanics*, 61 (5):543–558, 2018. doi: 10.1007/s00466-017-1535-8.

[50] L.-E. Lindgren. *Computational welding mechanics*. Elsevier, 2014.
[51] L.-E. Lindgren, A. Lundbäck, M. Fisk, R. Pederson, and J. Andersson. Simulation of additive manufacturing using coupled constitutive and microstructure models. *Additive Manufacturing*, 12:144–158, 2016.
[52] X. Lu, X. Lin, M. Chiumenti, M. Cervera, J. Li, L. Ma, L. Wei, Y. Hu, and W. Huang. Finite element analysis and experimental validation of the thermomechanical behavior in laser solid forming of ti-6al-4v. *Additive Manufacturing*, 21:30–40, 2018.
[53] A. Lundbäck and L.-E. Lindgren. Modelling of metal deposition. *Finite Elements in Analysis and Design*, 47(10):1169–1177, 2011.
[54] Marenostrum-IV website. https://www.bsc.es/marenostrum/marenostrum, (n.d.). Accessed: 2018-07-24.
[55] P. Michaleris. Modeling metal deposition in heat transfer analyses of additive manufacturing processes. *Finite Elements in Analysis and Design*, 86:51–60, 2014.
[56] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.
[57] M. Mozaffar, E. Ndip-Agbor, S. Lin, G. J. Wagner, K. Ehmann, and J. Cao. Acceleration strategies for explicit finite element analysis of metal powder-based additive manufacturing processes using graphical processing units. *Computational Mechanics*, pages 1–16, 2019.
[58] E. Neiva, M. Chiumenti, M. Cervera, E. Salsi, G. Piscopo, S. Badia, A. F. Martín, Z. Chen, C. Lee, and C. Davies. Numerical modelling of heat transfer and experimental validation in powder-bed fusion with the virtual domain approximation. *arXiv preprint arXiv:1811.12372*, 2018.
[59] L. Parry, I. Ashcroft, and R. Wildman. Understanding the effect of laser scan strategy on residual stress in selective laser melting through thermo-mechanical simulation. *Additive Manufacturing*, 12:1–15, 2016.
[60] N. Patil, D. Pal, H. Khalid Rafi, K. Zeng, A. Moreland, A. Hicks, D. Beeler, and B. Stucker. A Generalized Feed Forward Dynamic Adaptive Mesh Refinement and Derefinement Finite Element Framework for Metal Laser Sintering—Part I: Formulation and Algorithm Development. *Journal of Manufacturing Science and Engineering*, 137(4):041001, 2015.
[61] P. Prabhakar, W. Sames, R. Dehoff, and S. Babu. Computational modeling of residual stress formation during the electron beam melting process for Inconel 718. *Additive Manufacturing*, 7: 83–91, 2015.
[62] I. A. Roberts, C. J. Wang, R. Esterlein, M. Stanford, and D. J. Mynors. A three-dimensional finite element analysis of the temperature field during laser melting of metal powders in additive layer manufacturing. *International Journal of Machine Tools and Manufacture*, 49(12–13):916–923, 2009.
[63] T. M. Rodgers, J. D. Madison, and V. Tikare. Simulation of metal additive manufacturing microstructures using kinetic Monte Carlo. *Computational Materials Science*, 135:78–89, 2017. doi: 10.1016/J.COMMATSCI.2017.03.053.
[64] E. Salsi, M. Chiumenti, and M. Cervera. Modeling of microstructure evolution of ti6al4v for additive manufacturing. *Metals*, 8(8):633, 2018.
[65] W. J. Schroeder, B. Lorensen, and K. Martin. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.
[66] I. Setien, M. Chiumenti, S. van der Veen, M. San Sebastian, F. Garciandía, and A. Echeverría. Empirical methodology to determine inherent strains in additive manufacturing. *Computers & Mathematics with Applications*, 2018.
[67] J. Smith, W. Xiong, J. Cao, and W. K. Liu. Thermodynamically consistent microstructure prediction of additively manufactured materials. *Computational mechanics*, 57(3):359–370, 2016.
[68] J. C. Steuben, A. P. Iliopoulos, and J. G. Michopoulos. Towards Multiscale Topology Optimization for Additively Manufactured Components Using Implicit Slicing. In *Volume 1: 37th Computers and Information in Engineering Conference*, page V001T02A024. ASME, 2017. doi: 10.1115/DETC2017-67596.

[69] The Common Layer Interface (CLI) universal format. Common Layer Interface (CLI) Version 2.0 File Description, (n.d.). Accessed: 2016-08-26.

[70] T. Wohlers. *Wohlers report 2017*. Wohlers Associates, Inc, 2017.

[71] W. Yan, S. Lin, O. L. Kafka, Y. Lian, C. Yu, Z. Liu, J. Yan, S. Wolff, H. Wu, E. Ndip-Agbor, M. Mozaffar, K. Ehmann, J. Cao, G. J. Wagner, and W. K. Liu. Data-driven multi-scale multi-physics models to derive process–structure–property relationships for additive manufacturing. *Computational Mechanics*, 61(5):521–541, May 2018. doi: 10.1007/s00466-018-1539-z.

[72] W. Yan, Y. Qian, W. Ge, S. Lin, W. K. Liu, F. Lin, and G. J. Wagner. Meso-scale modeling of multiple-layer fabrication process in selective electron beam melting: Inter-layer/track voids formation. *Materials and Design*, 141:210 – 219, 2018. doi: https://doi.org/10.1016/j.matdes.2017.12.031.

[73] Y. P. Yang, M. Jamshidinia, P. Boulware, and S. M. Kelly. Prediction of microstructure, residual stress, and deformation in laser powder bed fusion process. *Computational Mechanics*, 61(5): 599–615, 2018. doi: 10.1007/s00466-017-1528-7.